Development of a Two-level Modular Simulation Tool for Dysim

Kofoed, J.E.

Publication date:
1988

Document Version
Publisher's PDF, also known as Version of record

Risø-M-2652

RISØ

Risø-M-2652

Development of A Two-level Modular Simulation Tool for Dysim

Jan Eggert Kofoed

Risø National Laboratory, DK-4000 Roskilde, Denmark
July 1987

RISØ-M-2652

DEVELOPMENT OF A TWO-LEVEL MODULAR SIMULATION TOOL FOR DYSIM

Jan Eggert Kofoed

Abstract. In this report a simulation tool that assists the user when constructing continuous simulation models will be described.

The simulation tool can be used for constructing simulation programmes that are executed with the runtime executive DYSIM86 which applies a modular approach. This approach makes it possible to split a model into several modules. The simulation tool introduces one more level of modularity. In this level a module is constructed from submodules that are taken from a library. A submodule consists of a submodel for a component in the complete model.

The simulation tool consists of two precompilers that work on the two different levels of modularity. The library is completely open to the user so it is possible to extend it. This is done by a routine which is also part of the simulation tool.

The simulation tool is demonstrated by simulating a part of a power plant and a part of a sugar factory. This illustrates that the precompilers can be used for simulation of different types of process plants.

July 1987
Risø National Laboratory, DK-4000 Roskilde, Denmark

ISBN 87-550-1330-9
ISSN 0418-6435

Grafisk Service Risø 1987

CONTENTS

Page

Page

PREFACE

This thesis is submitted for the partial fulfilment of the requirements for the licentiat degree (the Danish Ph. D.) at the Technical University of Denmark, Danmarks Tekniske Højskole, DTH.

The work was carried out at Risø National Laboratory at the Department for Energy Technology, and directed by Mogens Kümmel (DTH) and Poul la Cour Christensen (Risø). Also participating in our progress meetings were Henrik Søeberg (DTH) and Torben Petersen (Risø). I am very grateful for their cooperation and useful advice.

The simulation tool that has been developed was tested on very early stages in other works by Niels Larsen (Risø). He has made many useful suggestions and pointed out some of the errors that have been detected, and for that I am grateful.

At a late stage in the study Steen Weber (Risø) was trying hard to transcribe the simulation tool to a workstation, Tektronix 4404. During this period several violations of the FORTRAN77 standard were discovered. I am grateful to him for giving me this opportunity to improve the portability of the simulation tool.

Lars Bo Jørgensen at De Danske Sukkerfabrikker, Driftteknisk Laboratorium has provided me with valuable data and suggestions for modelling the multiple effect evaporator and I wish to express my gratitude for that help.

1. INTRODUCTION

In this chapter a short review will be given of what simulation is and how it can be used. The main focus will be on continuous system simulation since this is the type of simulation that is supported by the simulation tool which is going to be described in the report.

1.1. Computer simulation

The simulation of systems on computers is beneficial in many situations. These include, among other items:

- operator training
- optimization of systems
- study of hazardous accidents

A number of systems can be simulated, such as:

- chemical process plants
- power plants
- ecosystems
- electrical circuits
- financial systems
- shops

It comes as no surprise that these very different systems are likely to be best simulated using quite different approaches. A few years ago it was possible to categorize the approaches into two main areas: discrete and continuous simulation. Discrete simulation is used when the system is described by queues where items arrive and leave at schedules determined dynamically. The discreteness refers in other words to the time. In continuous systems the model description contains quantities that are given by differential equations so the time is continuous. Later these

approaches have been merged in some simulation systems and this is then termed combined simulation.

This project is concerned only with providing aids for simulation of continuous systems so the following discussion is related to only this type of simulation.

It is important to understand certain basic concepts when conclusions from a simulation are to be made. You will never be able to produce the results of the real world; the best you can hope for is to produce the predictions of a so-called base model (ZEIGLER, 1976). In a base model those components of the real world believed to contribute considerably to the behaviour of the system in question are included. Since the "reality" is basically distributed you will often take one further step in the process of modelling. This is to make a lumped model of the base model (see Fig. 1.1). In the lumped model some distributed parts are lumped together by assuming that all the properties in the distributed part (e.g., a tank) are the same independent of the position (i.e., there are no temperature gradients in a tank to mention one example). This lumped model is then described in some computer- or simulation-language. This constitutes the computer programme to be executed by a simulation programme. The results of this process, in the form of graphs, tables or whatever, represent the simulation. But the simulation of what?

The lumped model is continuous in time but the simulation programme discretizes this. Some effort must be put into assuring that the simulation programme uses a time step sufficiently small to produce results reflecting the mathematical model. When this has been done you know that the output represents the computer programme.

If the computer programme is without errors, i.e. represents the lumped model, one has a simulation of the lumped model. Now one is faced with the problem of validating the lumped model with respect to the base model. This is done by assuring oneself that the process of lumping was so "mild" that the behaviour of the

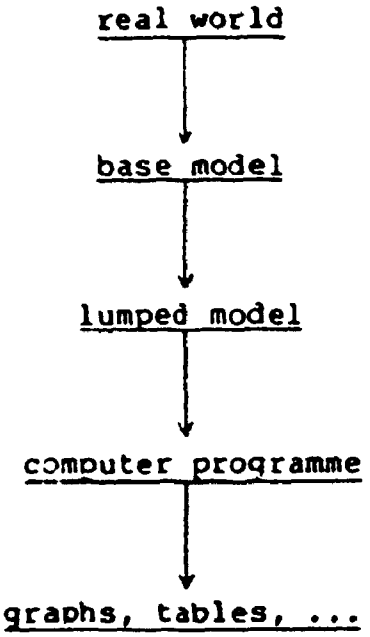


Fig. 1.1. Simulation, of what?

lumped model reflects "adequately" the behaviour of the base model.

Finally the base model must be validated with respect to the real world. This means that one must assure oneself that all the components needed to adequately describe the real world are present in the base model.

When all this has been done one can claim that the graphs, for example, show the results of a simulation of the real world as described in the base model. It is essentially necessary to understand each step in the simulation process to be able to judge the validity of the results.

1.2. Modelling and execution of the simulation

Here it will be discussed how one is going to model a process plant and later perform a series of simulations with the simulation programme. This is done in order to gain some knowledge

about the structure or nature of simulation programmes and we will also introduce a nomenclature to be used in the rest of the report.

A model of (a part of) a process plant consists of a set of differential and algebraic equations. The differential equations must all be of first order. A variable that is differentiated is called a state variable. A variable that is given by an algebraic equation is called an algebraic variable. These variables are called output variables since they are calculated in the model. They are calculated from input variables which are derived elsewhere, i.e. they are unaffected by this model. The process plant is described by some physical parameters such as volumes of tanks, tube dimensions, etc.

The first thing to do when a dynamic simulation of a process plant is going to be made is to make the programme implementing the lumped model as described above. This is done from diagrams of the plant.

Those parameters that are known from the start can be made part of the programme that is to simulate the process plant, while those parameters that are to be changed during the experiments which are to be performed on the model should not be made part of the programme; they are rather part of the information given at runtime. Assemblage of these parameters must be done on an early stage of the simulation.

The initial condition of the system must be given and this of course means the initial values of the state variables. Actually it also means the initial values of (some of) the algebraic variables since some of these may be used before they have been calculated. Any dynamic simulation of a system should start from a steady state, so firstly the steady state must be found. When it has been found it must be available when the disturbances are introduced.

After a transient (a disturbance of a steady state) has been run the result of the simulation is going to be studied. The result can be displayed as graphs, tables, etc.

The simulation system that is aiding the user must provide help for all the tasks mentioned above. This includes

- handling of initial conditions
- facility for steady state solution
- facility for transient solution
- plotting.

What makes good simulation systems differ from poorer ones are the quality of these facilities.

2. BACKGROUND

In this chapter the background on which the development of the precompiler system was performed is given. First the relevant information about the work at Risø's Section for Process Simulation is outlined and then a review of some of the work with simulation systems elsewhere is given. The work at Risø must be known because the precompiler system is an extension of the simulation software already available here. The works going on elsewhere must be known because this is where some inspiration can be gotten, and the present work must be evaluated from a knowledge of what is available elsewhere. It must also be known how the runtime executive is structured and working since it is going to control the simulations.

2.1. Work at Risø

In 1975 the simulation programme DYSYS (SCHLECHTENDAHL, 1970) was purchased as the starting point for digital simulation. In the first five years minor modifications were made to fulfil our needs but by 1980 it was felt to be necessary to make major modifications of DYSYS. Because of the programme structure it was hard to make these modifications in a reliable way and

Risø therefore made a new simulation programme: DYSIM (CHRISTENSEN, 1981).

Later it was recognized that a modular approach similar to that of SIMNON (ELMQUIST, 1975) and GPSM (BROWNE et al., 1986) would be desirable. The reasons for modularization on this scale are many (see for example GONZALES et al., 1984), but let us mention reduction of complexity and code maintainability. It must be noticed at this point that one may mean different things also when talking about modules. In the new version of DYSIM, in SIMNON and in GPSM a module is a section of the process plant; others, however, think of a module as representing a submodel for a component in the process plant. Examples of these are MMS (SCHAAK et al., 1986) and SPEEDUP (PERKINS and SARGENT, 1982).

In DYSIM which is written in FORTRAN77 there is a strict separation of the model description and the model experiment. By this is meant that the equations describing the model and the parameters that are not going to be affected during experimentation are separated from the instructions controlling the experiments to be performed on the model. The model description is written in a FORTRAN77 programme and the instructions needed at runtime, for the model experiment, are written in an input file.

In the new version of DYSIM (CHRISTENSEN, KOFOED and LARSEN, 1986) the modular approach was implemented by allowing the structure of the input file to be modular. This modular approach was extended with one more level, which is the use of submodels from a library with submodules. The submodels are typically models of components in the process plant, i.e. they are modules of the same type as in for example MMS as mentioned above.

This two-level modular approach to simulation is the basic concept of the precompiler system which was developed in this project. The precompiler system helps the user construct FORTRAN77 codes that model a process plant.

Until the end of 1986 the programmes were developed on a Burroughs B7800 mainframe computer but in December 1986 several

PCs were bought and the programme was also transcribed to these. The precompiler system works almost identically on the two computers but the runtime executive which performs the simulation and controls the input and output generation works in a batch fashion on B7800 while on the PCs it allows interactive control of the simulation.

2.2. Standards for simulation languages

In the mid-sixties several different digital computer routines had been developed to perform dynamic simulation of continuous systems. Due to a general feeling that some sort of direction for further development was needed an ad hoc committee was set up by Simulation Councils, Inc. (SCi). At the end of 1967 the result was published (STRAUSS (editor), 1967). A continuous system simulation language (CSSL) is proposed and time showed it to be a great success. For 15 years this was the only attempt to standardize simulation languages. This proposal will be referred to as CSSL67. Two committees then started working on a new standard. One is established by the International Association for Mathematics and Computers in Simulation (IMACS) and it is known as Technical Committee 3 (IMACS/TC3), the other is the successor of the original SCi committee, the SCS technical committee on CSSLs. These two committees wanted a draft proposal for further discussions and a group at Salford University made one, the CSSL81. This draft was so interesting to the European Space Agency that they ordered a simulation language along these lines. The CSSL81 proposal is outlined in CROSBIE and HAY, 1982.

Both of these two proposals were made in order to get a standard and both standards seek features such as clarity in model definition, open-ended systems, sorting of statements, ability to use routines written in other languages and ability to perform experiments on the model.

The CSSL67 proposal initiated the development of a series of simulation languages. The one that is the most widespread is probably the Advanced Computer Simulation Language: ACSL (MIT-

CHELL and GAUTHIER, 1986) which along with most other simulation languages following the CSSL67 proposal has inherited the ideas from simulation on analog computers. Of course this made it easier to replace an analog computer with a digital and perform the simulation on it. Actually one of the application areas for the first digital simulation programmes were the testing of the simulations performed on analog computers (STRAUSS (editor), 1967). Today this way of thinking is old fashioned.

Most of the simulation languages based on the CSSL67 proposal are translating the user's programme to FORTRAN. This makes it possible to use FORTRAN routines in the simulation programmes which is an advantage since FORTRAN is (or was!) used extensively. It is a disadvantage though that the runtime error reporting becomes inadequate because this is done on the FORTRAN level. Because of the sorting routine in the simulation systems that follow the CSSL proposals the FORTRAN version of the simulation programme may only barely resemble the user's programme. The use of macros which is part of the proposals only makes this worse. When a macro is used a set of statements will replace the macro and this of course may not be statements that the user is familiar with. Furthermore it demands that the user should be familiar with FORTRAN to be able to understand what was the cause of the runtime error.

It was hoped that some of the problems mentioned above might be removed by the CSSL81 proposal. The intentions were to maintain some continuity from the CSSL67 on the one hand but on the other hand some radical changes were needed. The emphasis was now put almost entirely on continuous systems even though the efficient handling of discontinuities is considered very important. Also the ideas of structured programming as they are implemented in ordinary programming languages such as PASCAL has influenced the CSSL81 proposal.

CSSL81 proposes a separation of the model from the simulation experiments to be performed on it. This should be made in a fashion that allows several experiments to be performed on the same model and the same experiment to be performed on several models.

The notion of submodels is made as opposed to the macros of CSSL67. Instead of inserting the "submodel" at each place it is used in the model (as it is proposed in CSSL67) the submodel is called from the model where it is used and the exchange of information is made by passing input and output variables as arguments to the submodel. This greatly impedes the sorting of statements. Some restrictions must be put on the structure of the submodels but the advantages gained in the use of submodels versus the use of a sorting routine justifies the inconveniences of using submodels (CROSBIE and HAY, 1982).

The optional segmentation of the model in CSSL67 makes it possible to use several integration routines on a sequential processor while the segmentation in CSSL81 furthermore provides a means of using parallel processors.

In summarizing CSSL81 differs from CSSL67 by

- almost entirely supporting continuous systems with discontinuities;
- using structured programming techniques including mandatory declaration of variables and (almost) avoiding the "nasty" GOTO statement;
- separating the model from the simulation experiments performed on it;
- the usage af submodels;
- segmentation that allows parallel processing.

A few of the problems not addressed directly in CSSL81 is a discussion of the range of data types to allow access to and a proposed strategy for handling algebraic loops with several variables.

As was mentioned in the beginning the European Space Agency had a simulation language made on the basis of the CSSL81 proposal. It is called ESL: European Simulation Language (CROSBIE et al., 1985). The automatic sorting that already has been mentioned as a problem in CSSL81 has not been implemented in ESL.

In any attempt to set a standard it must be recognized that despite every good intention everyone will be biased toward one's own simulation language (RIMVALL and CELLIER, 1986) and therefore it seems practically impossible to settle on a particular standard. This also is believed to reflect the fact that many demands are placed on a simulation language, varying from one application to another. Often they are more or less mutually in conflict, as for example the discussion concerning modularity and submodels versus sorting of statements. This point is for example illustrated by Wood's criticism of CSSL81 and ESL (WOOD, 1986).

We have now outlined the background for the project at Risø and given a short review of the main ideas of simulation systems in general. With this project it was decided to investigate ways to facilitate the phase of programming of continuous models by developing an interface to DYSIM86. Using this interface it should not be necessary to know the structure of DYSIM86 but rather the interface should take care of that error prone work. The interface should also allow the re-use of programmes modelling components. It should not be so that only a certain type of process plants can be simulated but rather that each user gradually develops a set of component models that allow quick and reliable programming of complete models of process plants. The interface is a programme that compiles some files prepared by the user. In order to be able to describe the precompiler system we must first outline the background concerning the software that the precompiler is going to be an interface to.

2.3. DYSIM86

The runtime executive in the modular version of the simulation system DYSIM is called DYSIM86. This has been described elsewhere (CHRISTENSEN, KOFOED and LARSEN, 1986) but since the precompiler system is developed to facilitate simulation using DYSIM86 it is necessary that the structure is shortly outlined here as well.

A schematic description of a simulation programme linked with the runtime executive and the auxiliary files with initial conditions, output, etc. is shown in Fig. 2.1. The runtime executive manages initialization and integration of the model, preparation of output to prints and plots and, if necessary, the administration of the large data buffer that is used to simulate a pure time delay.

Initialization and control of integration as well as the setting up of output is made on the basis of an input file. The input file contains all the runtime information. A list file makes it possible to use symbolic names for the model variables in input, output, and when performing the simulation.

The model will typically consist of several modules and this modularity is reflected in the input file and the list file. The modules are assembled in a connecting system to make up a total model of the process plant under investigation. DYSIM86 communicates with the model through the connecting system. The precompiler system which is the issue of this report is a tool that

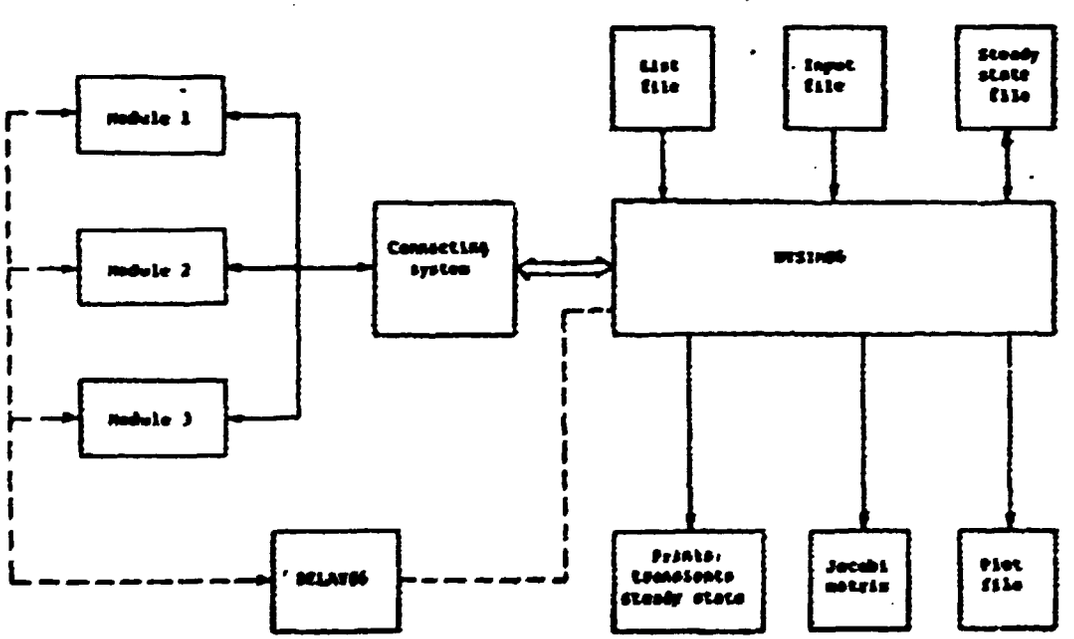


Fig. 2.1. Schematic description of a simulation programme using the runtime executive DYSIM86.

shall facilitate the construction of the model, i.e. the modules and the connecting system. Therefore we will not describe the input file and output form of DYSIM86. This is described elsewhere (CHRISTENSEN, KOPOED and LARSEN, 1986).

A module is written in a FORTRAN77 subroutine and it contains a model of a delimited section of the process plant. Testing of the complete model is facilitated by testing each module individually. When this has been done the concept of DYSIM86 supports easy access to assemblage of the modules to make the final testing and followed by execution of the interesting simulation experiments.

The connecting system manages the transfer of variables to and from the runtime executive DYSIM86 on the one hand and the modules on the other. Communication with DYSIM86 is made through common blocks and the user will probably find it convenient also to use common blocks for the communication with the modules.

Depending on the chosen integration routine DYSIM86 calls the model, i.e. the connecting system, one, two or more times for each integration step. An integration step can be rejected because of accuracy control or by the user and therefore the state of the system at the start of an integration step must be available for replacement of a rejected step. This means that two sets of variables must be used, namely the new variables and those that may be going to replace them if a step is rejected. Transfer from one set to the other is most efficiently made using arrays to hold the variables. This greatly blurs the meaning of the variables and leads to many errors. Furthermore the user must be familiar with the times at which to transfer variables between the two sets.

These are all some details that are concerned with DYSIM86 and with FORTRAN77. They have really nothing to do with the task of performing simulations. By studying the FORTRAN programming language and the runtime executive DYSIM86 this can of course be done but it is very probable that it would be done with occasional errors. A routine knowing how to manage the communica-

tion will eliminate several of the errors that were made previously. The routine will take care of the communication and it will make it possible to use symbolic names for the variables in the connecting system instead of using elements of arrays. The symbolic names are those used when the modules were made, and they are the same names that shall be available to DYSIM86 through the list file.

This routine is part of the precompiler system that we are going to look at. A syntax different from the standard programming language FORTRAN77 is necessary when the use of symbolic names must be possible. The syntax to settle on depends on the facilities that are desired to be handled by the precompiler system. This will be the issue of the next chapter.

3. DEVELOPMENT OF THE PRECOMPILER SYSTEM

In this chapter it will first be described what the concept of the precompiler system is. The precompiler system produces some FORTRAN77 files from some source files that have been prepared by the user. The source files must be written according to some syntax which will be described and a short note on the reason for the particular choice may also be given. The syntax is described by listing the syntax for the different types of commands followed by defining how the three different types of source files must be structured.

At the end of the chapter it will be described how the source files may be corrected interactively by the precompiler system when an error is being detected. Also a special notation concerning the connection of modules and (especially) submodules is going to be described here. This notation was used when the model of a part of sugar factory was developed.

3.1. Introduction

The usage of precompilers as a tool that facilitates construction of simulation programmes is widespread. They are used for all types of simulations. For example, one has been developed for the General Purpose Simulator System (GPSS) (LAM, 1978) which is a system for discrete simulation. The impact from the committee making the CSSL67 proposal is seen by the fact that there has also been made a precompiler that constitutes a CSSL interface to GASP IV (BIRTA and KHADR, 1979) which is a simulation system for combined systems, i.e. systems that are both continuous and discrete. An example of a precompiler that is being used for simulation of a special type of process plants is MMS-EASE+ (SCHAAK, BARTELLS and LONG, 1986).

The precompiler system that is going to be described in this chapter is of the general type, i.e. "any" continuous system

can be simulated. The precompiler system is going to support simulation using the runtime executive DYSIM86 which is able to handle any continuous system. MMS-EASE+ supports simulation using the simulation language ACSL which also is of the general type so the restriction in applications with MMS-EASE+ mentioned above is within the precompiler itself.

3.2. Concept for the precompiler system

When defining the concept for the precompiler system it must first be decided what one expects from the precompiler system. Of course it should speed up the construction of reliable simulation programmes for DYSIM86. This can be done in at least three ways. Firstly by relieving the user from remembering a lot of the details that is not related to the model but only to the execution of the simulation. Secondly by allowing the use of well-tested submodels for construction of modules. Thirdly by allowing the use of symbolic names in the connecting system. Of course any other facility is welcomed.

Since the precompiler system is tangled to DYSIM86 it inherits the modularity of this runtime executive. As is proposed in CSSL81 (CROSBIE and HAY, 1982), the well-tested submodels are placed in a library. A complete model of a process plant is therefore built in two stages (see Fig. 3.1):

1) assembling submodels from a library to create a module;
2) assembling the modules to create the complete model.

We therefore say that we are using a two-level modular approach. In the figure it is also seen that the precompiler system handles the precompilation of the library. In principle this is only necessary once.

It was decided to make the modular approach absolutely strict in the sense that a submodel is not allowed to use another submodel and a module is not allowed to use another module. The reason for this is that the modules and submodules are going to be FOR-

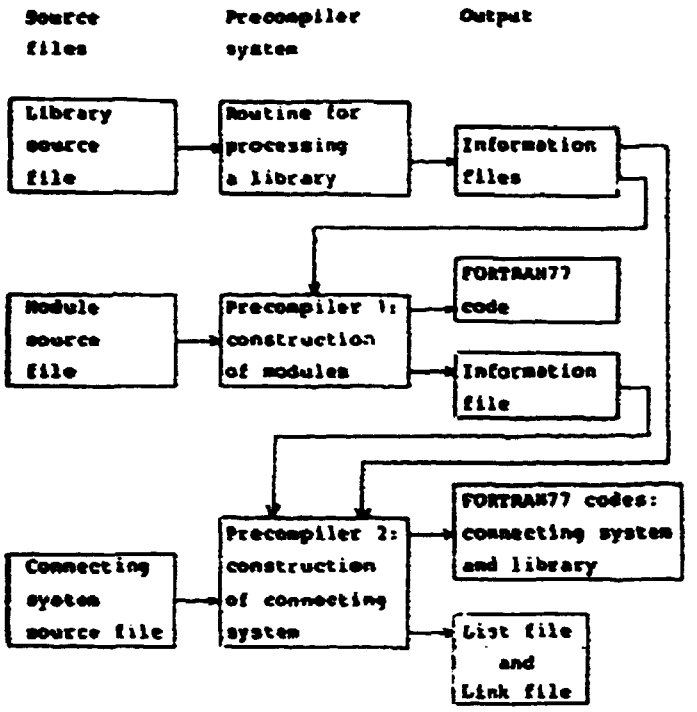


Fig. 3.1. The precompiler system and the input and output files shown schematically.

TRAN subroutines and in the FORTRAN77 standard (ANSI, 1978) recursion or nesting is not allowed. It is possible to avoid recursion with milder restrictions than the one chosen but it is the simplest way and it is not thought to be a serious restriction. In the CSSL81 proposal (CROSBIE and HAY, 1982) and also in ESL (CROSBIE, 1986) nesting of submodels is allowed.

The constructions of programmes in modern, ordinary computer languages should be available and one simple way to do this is to just use one of these languages with a few extensions. In other words it was decided not to create a new simulation language but merely extend the set of executable statements in FORTRAN77 with precompiler commands. FORTRAN77 was chosen because eventually the complete model of the process plant must be constructed in this language. The task of creating a new simulation language would extend beyond the time limit of a Ph.D. study. There are other reasons which will be given in the following discussions.

It is recognized that a sorting routine is a desirable facility and this is also seen by its retention in the CSSL81 proposal. The use af modules makes sorting hard (CROSBIE et al., 1985) and using a two-level modular approach will make it even harder. No attempts to sort the statements were made in this project and it is therefore the user's responsibility to write the statements in the correct order. "The correct order" is not always attainable; the presence of algebraic loops makes this impossible. This issue is discussed elsewhere, 6.1.2 "Pros and cons of modularity".

The library must be open-ended making it possible to extend the library with well-tested submodels. It is desirable to make it easy to test and modify a submodel as a module and thereafter append it to the library as a submodule. This is accomplished by choosing a convenient syntax. It will be seen that the syntax of a submodule is exactly the same as that of a module. Therefore it is possible to programme a submodel as a module which is easier to test and when the submodel has been validated the module can be appended to the library as a submodule in a simple way by just changing the name-giving command.

It was decided to let the precompiler system work on source files prepared by the user with a standard editor on the computer. The process of precompilation is done interactively. The user will be prompted for commands and files with source text and will be told the names of the relevant files that are produced by the precompiler system. If a syntax error is found by the precompiler system the user is given the opportunity to interactively correct the source file. This relieves the user from terminating the session, edit the source file, and resume precompilation. This is of course possible only for certain types of errors. It has been restricted to be those errors where it may be possible to correct the error at the line being currently processed.

In summarizing, the following decisions were made:

1) apply a two-level modular approach;
2) use the executable FORTRAN77 statements;
3) extend with precompiler commands when constructing the source files;
4) make the library open-ended;
5) do not attempt to sort the statements; and
6) provide interactive corrections on a line-to-line basis.

3.3. Choice of syntax

The reasons for choosing a syntax is to supply the precompiler system with the necessary information and to make it possible to perform the precompilation. By "the necessary information" is meant the names of the variables carrying information, for example; by "making precompilation possible" is meant facilities that allow use of submodules in the modules, assemblage of modules in the connecting system and the use of symbolic names in the connecting system.

It is possible to use FORTRAN77 comments (lines with an asterix "*" in column one) in the source files but there will also be provided for precompiler comments. A precompiler comment is a line with an ampersand "&" in column one. Such a line is completely ignored by the precompiler system.

3.3.1. General remarks about commands

It must be possible for the precompiler system to distinguish commands from FORTRAN77 statements. This is done by using a special character in column one of a precompiler command line. This character is chosen to be an exclamation mark "!". The name of the command is written after this symbol. If additional information is needed on the same line it follows the name which is succeeded by (at least) one blank character. Often it is not possible to write all the information on the same line and therefore it must be possible to continue on the next line. This is done by writing an exclamation mark in column one of the continuation line, optionally writing the name of the com-

mand, adding (at least) one blank and then the additional information. (For an example see Fig. 3.3).

There are mainly three groups of commands. They are

1) commands for naming modules and submodules
2) commands for naming communication variables
3) commands for implementation of modules and submodules.

Beside these a few more precompiler commands can be used for special purposes.

The connecting system differs from modules and submodules since there can be only one connecting system. Therefore it is not named with a command. Instead a command supplies information about which modules are being used in the connecting system.

One more command needs to be commented on. It is a command that can be used at the same location as those for naming variables and it follows exactly the same syntax. The purpose of the command is to supply the precompiler system with information about which functions from the library that are being used. The functions must be present in the same library as the submodules.

The three groups of commands differ in their purpose and also in their content of information, and therefore they will necessarily differ in syntax. In the next subsections the commands will be described briefly.

3.3.2. Name-giving commands

Since there can be several submodules and modules each must be uniquely named. Supplying a name is done by writing the name-giving command in the first line (that is not a comment). The information contained in the command is merely a name of the module or submodule and this is simply written right after the name of the command. Examples of the two name-giving commands are

```
!SUBMO PICON     and
!MODUL  TANK
```

for naming a submodule and a module. These commands cannot be continued on several lines.

The names of the submodules and modules become the names of FORTRAN77 subroutines and this imposes a limitation on the length of the names as well as on the set of characters that can be used in a name. The length of identifiers in FORTRAN77 is not allowed to exceed 6 and the set of allowed characters contain all alphanumeric characters and all digits with the explicit exception that the first character may not be a digit. These limits of FORTRAN77 are also the limitations on the names of the submodules, but the length of names of modules is restricted to be less than or equal to 3. This limitation is inherited from the runtime executive DYSIM86.

The name-giving command is mandatory.

3.3.3. Type-giving commands

Communication is carried by variables. The names of these variables must be known to the precompiler system and therefore they are written in some commands. There are several types of variables and therefore there are several type-giving commands. There are four types of commands that are related directly to the communication concerning the numerical calculations of the simulation. A string variable (a text) can also be used to transfer information to and from a submodule and to and from a module. One more command will be mentioned in this connection. It is not used for supplying information about variables and it is therefore not truly a type-giving command, but since it follows exactly the same syntax and is located at the same place it will be included in this subsection. The command is used for supplying the precompiler system with information about which functions are being used. The five type-giving commands together with this additional one are given in Fig. 3.2.

The information in the type-giving commands are the names of the variables. Since there can be many variables it will often be necessary to use continuation lines. The names of the variables are written after the name of the command and a comma "," is used for separating one variable from another. A semicolon ";" terminates the command. If a line is to be continued the last nonblank character must be the separating comma.

output	input
STV state variables	INP input variables
ALG algebraic variables	PAR parameters
TXT string variable (not in the connecting system)	
FUN functions	

Fig. 3.2. The five type-giving commands plus the FUN command.

The variables can be either simple variables or one-dimensional arrays (tables). Names of variables are just an identifier following the FORTRAN77 standard (see subsection 3.3.2 "Name-giving commands"). Names of one-dimensional arrays are an identifier followed by the dimension enclosed in parentheses.

There are a few restrictions. String variables cannot be transferred to the runtime executive DYSIM86 and therefore it is not allowed to use the TXT command in the connecting system. With the TXT command it is only allowed to transfer one variable. Variables transferred to and from a submodule are restricted to be simple variables. A simple example of a STV command from a module or the connecting system is given in Fig. 3.3.

```
!STV    A,B(5),
!       C,D1;
```

Fig. 3.3. An Example of a STV command (in a module or in the connecting system).

Everything that has been said about the syntax for type-giving commands are also true for the FUN command except of course that the concept of "arrays" is nonsense in this connection. The names of the functions follow the FORTRAN77 standard for identifiers.

Only the necessary type-giving commands must be written. The order of the type-giving commands is subjected to no restriction.

3.3.4. Connecting commands

In a module several submodels can be connected with each other. The submodels can be either models of components written directly in the module using standard FORTRAN77 statements or submodules from the library. This is the first level in the two-level modular approach that has been applied for the precompiler system. At the second level modules are connected to each other in the connecting system. These two different types of connections are made by the two connecting commands MACRO (in a module) and MOD (in the connecting system). In both cases the implementation is made using a CALL statement but the transfer of variables is managed differently because a module can be used only once while a submodule can be used several times. This is the reason for the big differences between the two commands.

Since a submodule can be used several times all the variables containing information must be transferred back and forth each time a submodule is used in a module; this is done most efficiently by passing them as part of a subroutine argument list. With the MACRO command there will be transferred variables from the module to the local variables of the submodule. This is done

by writing the matching pairs of variables from the module and the submodule in a series of subcommands. A sub-command is used for starting and one is used for terminating the connecting command.

The first sub-command is the MACRO sub-command which is of exactly the same form as the name-giving commands since this is precisely what it does: it supplies the name of the submodule that is going to be implemented.

After the MACRO sub-command there follow several connecting subcommands which take care of establishing the connection between the pairs of variables from the module and the submodule.

The names of the connecting sub-commands are the same as those of the corresponding type-giving commands. The part of the subcommand containing the information that must be supplied to the precompiler system is a series of pairs of variables. The pairs are separated by commas "," and the subcommand is terminated by semicolon ";" and continuation lines are allowed in exactly the same way as for type-giving commands. (For an example see Fig. 3.4).

```
!MACRO NAME
!STV    Z:Z(2) , Y : Y1;
!PAR    TAU=10.0,
!PAR    PB=0.50;
!END
```

Fig. 3.4. A constructed example of a MACRO command.

The pairs of variables establish connections between variables in the submodule and the corresponding variables in the module. The pairs are written with the submodule variable first followed by the connecting character colon ":" and then the module variable. Parameters in the PAR sub-command make an exception to this and the reasons for this will be discussed.

When a model of a section of a process plant is made there will be certain parameters of the components that will be given in advance. These parameters are constants and therefore they can be placed in the module without introducing any inconveniences for the user, in fact it is felt that they are placed at the appropriate place namely where they are used. All the system parameters that are likely to be changed during a simulation or between two simulations should be placed outside the simulation programme. This facilitates the e ecution of the simulations and the documentation.

In the PAR sub-command the module variable is replaced by the constant real parameter and an equals sign "=" is used instead of a colon.

It is understood that exactly those connecting sub-commands that correspond to the type-giving commands in the submodule are expected and that all the corresponding submodule variables in the given type-giving command must be referenced in the connecting sub-command. There is no restriction on the order of the connecting sub-commands or on the order of the pairs within a sub-command.

The MACRO command is terminated by an END sub-command which contains no additional information.

A simply constructed example of a MACRO command is shown in Fig. 3.4. It is intended only to give examples of the structure and of the two different types of connecting sub-commands.

By using the connecting command MOD in the connecting system the modules are connected to each other and are supplied with input. A module is used a' only one place in the connecting system and therefore only those variables that have been changed since the last call of the module need be updated. The precompiler "knows" how the runtime executive DYSIM86 operates and takes care of everything except those variables that are used for connecting one module with another and for introducing changes or disturbances. Input variables are used for these purposes.

Connections between modules are established by transferring an output variable from one module to an input variable of the other. This can be done in a simple assignment statement. Disturbances can be introduced by changing the value of an input variable. This can be done in several ways but step changes and linear ramps are the most common. Disturbances can most easily be controlled with parameters to the connecting system.

Because algebraic expressions should be allowed it was decided to use ordinary assignment statements for establishing connections and introducing disturbances. Furthermore, it has been seen that the user has the responsibility of supplying only information about the input variables in a module. The MOD command will then contain two sub-commands for bracketing the assignment statements establishing the connections. The first sub-command is the equivalent of the MACRO sub-command and here it is called the MOD sub-command. The last sub-command is exactly the same as for the MACRO command: the END sub-command.

In the connecting system symbolic names for the variables can be used. By this it is meant that the name of a variable from a module can also be used in the connecting system. To avoid problems when two different variables from two different modules have the same name a unique extension is added to the name of the variable, but this will be discussed in the next subsection where also important and necessary information about the MOD command is found so it must be read to get a full understanding of how to use the command properly. An example of a simple MOD command is given in Fig. 3.5.

```
!MOD    M2
        I1=O1.REG
        I2=O2.M1
        I3=S1.CON+AMIN1(1.,T/RT.CON)*(S2.CON-S1.CON)
!END
```

Fig. 3.5. A simple example of MOD command.

Since communication through variables is a way of representing the transfer of information from one module to another, in this case, it is clear that an input variable should be calculated on the basis of output variables. No input variables should be present in an expression in which an input variable is calculated. Also, connecting system parameters are allowed since this is the natural way to programme disturbances as noted above. This is just one example of the type of check for conceptual error detection that is being performed by the precompiler system. This type of error detection is by no means as thorough as possible and desired.

3.3.5. Symbolic names

First let us ask ourselves why symbolic names may be needed or at least may be desirable. Symbolic names come in handy because they let us choose names that makes us think immediately of the property in the process plant that is being represented by this variable. There is nothing mysterious in this; it is exactly what one does in every programme written in an ordinary programming language. Therefore we have easy access to usage of symbolic names since the precompiler system is translating the user's source text into FORTRAN77.

In the submodules and modules the variables are used exactly where the user is writing them and only there, but this is not the case for the connecting system as was discussed in the previous subsection. In the connecting system the precompiler handles the additional transfer of variables between two sets of arrays holding the values of the variables. One of the great advantages gained from using the precompiler system is that the user does not have to use the arrays when a variable is needed. Instead a syntax for naming variables has been chosen allowing the use of symbolic names. In the connecting system it must be possible to use the same names for the module variables as those that were used when the modules were constructed. It must also be possible to use the same name for "different" variables in different modules (in actual fact it is advisable to do so). Therefore a way of distinguishing them must be applied. This

is done by giving each variable from a module a unique extension, chosen to be the unique name of the module. Because variables that are transferred to and from the connecting system should also be available through the use of symbolic names, a unique extension for the connecting system must be found. As has been noted earlier each module, as well as the connecting system is translated into a subroutine. The names of the subroutines with the modules are the names of the modules, and the name of the connecting system is CON which is determined by the runtime executive DYSIM86. This makes it a natural choice to use CON as the unique extension for the symbolic names for variables in the connecting system.

The unique construction of symbolic names for variables in modules and in the connecting system is then chosen to be first the local variable name immediately followed by a period "." and either the module name or CON for the connecting system. (For examples see Fig. 3.5).

If a simple variable is used it is easy to construct the symbolic name for the variable, but how is an element of an array referenced? The extension is as for the simple variable, of course, but the part referring to the array element in the module is chosen not to be written as it is referred to in the module. When the element is used in a module it is written according to the FORTRAN77 standard, i.e. as the array name followed by the number of the element in parentheses. In the connecting system the parentheses are not written. Unfortunately, this introduces a slight limitation on the choice of names in the modules. If an array ARR in the module with 5 elements is being used the fifth element (for an example) is referred to in the connecting system as ARR5 and therefore it is not allowed to use a simple variable with the name ARR5.

This unique construction of variable names is exactly the same as the one chosen in the runtime executive. Therefore these names are those that will be used in plots and tables and during a simulation.

There is one exception to the use of symbolic names for variables and it has been introduced for the convenience of the user. It can be done because in one particular instance it is known to the precompiler system where to look for the variable. It is related to the MOD command where the connections to a module are established. The connections are made through the input variables of the module and only the input variables of the module are currently being connected, so that it is they that can be referenced. In simple assignment statements it is not necessary to write the extension for the input variables. With this additional information the MOD command is fully described as promised in the previous subsection.

3.3.6. Differential equations

The dynamic behaviour of a continuous system is described by differential equations. In the CSSL67 proposal (STRAUSS (editor), 1967) differential equations had to be transformed into integral equations and had to have the initial conditions given in the programme. This is inherited from analog simulation and it seems to be old-fashioned. In ESL (CROSBIE et al, 1985) which is based on CSSL81 (CROSBIE and HAY, 1982) another notation is being used. Now the differential equation is written by introducing the differential quotient of the state variable and writing the value just as if it was an algebraic equation. The same concept was (independently) chosen for the precompiler system, but in a fashion that is not as natural. It was chosen not to introduce a notation for the differential quotient of the state variables but instead to indicate the differential equation by using a colon ":" instead of the equals sign "=" in algebraic equations.

In the very beginning of the project a variable for each state variable was introduced, to which the value of the differential quotient should be assigned. The user had to name these with a precompiler command. This way of handling the differential quotients was obtained from SIMNON (ELMQVIST, ÅSTRÖM and SCHÖNTHAL, 1986) but soon it was found to be unnecessary and the precompiler was made to create variables with the differential quo-

tients automatically. The names of these variables are in the submodules constructed by appending a three-digit number to the letters "DZZ". In the modules and connecting system the differential quotients are stored in arrays constructed from the module name. This introduces a limitation on the names for variables that can be chosen by the user.

It can be seen that only first-order differential equations can be solved and every higher-order differential equation must be transformed into several of first order. This is true of course in other simulation systems also. Some allow use of higher-order differential equations through the use of transfer functions but only with some limitations in, for example, initial conditions. An example can be found in the ACSL manual (MITCHELL and GAUTHIER, 1986) where it can also be seen that the transfer function is transformed into several first-order differential equations.

3.3.7. Commands for special computations

The runtime executive DYSIM86 needs at least one and maybe two subroutines where the user can perform some special computations. One of these subroutines is called when a simulation is terminated. Here the user can perform final calculations and printouts. This subroutine is mandatory. Also the user can instruct the runtime executive to "re-call" the model when a time step has been accepted and in this case yet another subroutine must be present. If this facility is not used the subroutine need not be present. Both subroutines are expected to have a given special name for the runtime executive to be able to use them.

The precompiler system places the two subroutines as entry points in the connecting system. (In many respects an entry point is similar to a subroutine (ANSI, 1978).) The user need not write anything unless the special computations are needed or desirable and in this case two commands in the connecting system can be used.

The OUT command is used when computations and printouts at the termination of a simulation are wanted. The OUT command marks the start of the entry point. The entry point is terminated by either starting the next entry point with the DYN command to be described or by terminating the source file. Because it is an entry point in the connecting system all the variables in the connecting system are available and even the symbolic names can be used since the precompiler system also processes this part. A point should be noted: as implied earlier (in 3.3.5 "Symbolic names") only simple variables and elements of an array can be referenced, if all the elements of an array are needed every element must be written individually.

The DYN command marks the start of the entry point used for calculations to be performed by request of the user in his model. When the current step has been accepted this entry point is called by the runtime executive. The remarks above about symbolic names are also true here. The entry point is terminated either by starting the next entry point with the OUT command or by terminating the source file.

The user will probably find that it is convenient to perform the special computations not only in the entry points in the connecting system but also those in the modules. In this case the FORTRAN77 CALL statement must be used. A quite simple but very powerful example is given in both of the two simulation examples in Appendices C and H. If more sophisticated uses are wanted information must be sought elsewhere (ANSI, 1978).

Further details about these two entry points must also be sought elsewhere (CHRISTENSEN, KOPOED and LARSEN, 1986).

The names of the submodules, functions and the modules must be unique and they must be different from those that have been used in the runtime executive DYSIM86. A list of names that are reserved to DYSIM86 must be consulted if such a problem occurs.

3.4. File structure

The precompiler system uses a set of files. The user has to know the names of the files that are being constructed in order to avoid conflicts in name-giving. It is only necessary for the user to concern himself with a few of these file. The reason for using so many files is that this makes it possible for the precompiler system to support the user in the most smooth way. There can be said to be three kinds of files:

1) source files which are made by the user
2) FORTRAN77 files which are made from the source files
3) information files.

Not categorized above is the list file produced by the precompiler system. This section will describe the different types of files and will describe and give the grounds for why this file structure is convenient.

3.4.1. The concept

Files can be used for storing information that has been extracted from a source file and that is convenient to have access to at a later time. In this way the precompiler system need not prompt the user for that information. For each source file that is produced by the user the precompiler system will produce one or more files. By constructing the names of these files from the names of the source files or by other means the user need not name these files. This has another (and more important) advantage: the precompiler knows exactly what the names of the files are and that no situations will arise where the user uses a wrong name by mistake.

There is one more way to relieve the user from typing names of files. There is only one connecting system and today it is only allowed to use one submodule library so by choosing default names for these source files the user need not type these names. It is possible to use names other than the default names, however.

Finally one more choice for default names is made. Since each source file causes one or more files to be produced it is convenient to be able to tell which files that belong together. This is done in the way the precompiler constructs the names of the "derived" files. Each family of files derived from a source file will have the same name but the individual files will have different extensions added. The extensions produced by the precompiler system uniquely identifies what type of file it is. The default for source file extensions has been chosen to be SOU and if the user has followed this default it is not necessary to write the extension.

The discussion about extension to file names really applies only to the PC version of the precompiler system. On the mainframe version the file names follow another syntax where the term extension cannot be used directly. On the mainframe the file names are constructed in several levels. These levels can be said to be the equivalent of the directory structure of the Disk Operating System (DOS) on the PCs. It is beyond the scope of this report to give a detailed discussion of this subject, but it can be stated briefly that the last level of the mainframe name structure is made the equivalent of the extension in DOS.

On the mainframe computer one more piece of information is needed. This is also caused by the differences in the structure of file names. Using DOS on the PCs all the files that are being used for a specific simulation are placed in a directory and all file names refer to this directory. This is not the case on the mainframe, but it must be possible to work with different simulations at the same time so the files must be separated in one way or the other. This is done by choosing a system name for each system that is being simulated. This system name is then made the first level in the file names on the mainframe.

It is important to understand the construction of file names to avoid losing files when a new file is produced. This is especially a problem for source files with a module, see 3.4.3 "A module". Fig. 3.1 illustrates schematically the precompiler system including the set of files that are being produced.

3.4.2. The library

Presently it is allowed to use only one library in which the available functions and submodules are assembled. Because of this it is possible to choose a default name and the choice is SUBMOLIB. The name of the source file extension should furthermore be chosen by the user to be SOU so that it is possible to supply the precompiler system with the name in the most convenient way.

The information needed from the library is extensive and of different kinds; therefore the files that are produced by the precompiler system are many.

First an exact copy of the source file is produced in the PC version. From this file the names of the input and output variables of the submodules are found. When seeking this information the precompiler system should be able to go directly to the line with the relevant data. It is waste of space to copy the whole file to do that, but it was done nonetheless to save time during development. The first version of the precompiler system was programmed on the mainframe computer and there it does not matter what type the file is (sequential or direct access); it is possible to use direct access in any case. This is non-standard and not possible on the PCs. It was found to be time saving to just make an exact copy of the library source file but in a direct access version. The extension for the file is RAC (Random ACcess) and it is produced when using the precompiler routine for processing a library.

Precompiler 2 will be seen to be able to tell exactly what functions and submodules from the library that have been used in the modules and in the connecting system. To avoid compiling a large library the precompiler system will pick out only the necessary parts of the library and place them in a FORTRAN77 file. The extension for this file is FOR.

In order to translate the source file to FORTRAN77 only once, the submodules in FORTRAN77 code are placed in a file with extension F77 by the precompiler routine for processing a library.

This is where precompiler 2 picks out the submodules used in the system.

At every stage the precompiler system must know where to look for the necessary information; therefore an information file is produced by the precompiler routine for processing a library and the extension of this file is INF.

An example of a small library is listed in Fig. 3.6.

```
A:DYSIM\OPG\SUBMOLIB.SOU                02/26/87
-----------------------------------------------
      1       FUNCTION ALIM(XMIN,X,XMAX)
      2       REAL ALIM,X,XMAX,XMIN
      3       ALIM=AMAX1(XMIN,AMIN1(X,XMAX))
      4       RETURN
      5       END
      6 CCCCCC      ******
      7       FUNCTION ALIMD(Y,XMIN,X,XMAX)
      8       IF((X.GT.XMIN) .AND. (X.LT.XMAX))
      9      #  THEN
     10            Z=Y
     11       ELSE IF(
     12      #  ((X.LE.XMIN) .AND. (Y.LT.0.))
     13      #  .OR.
     14      #  ((X.GE.XMAX) .AND. (Y.GT.0.)))
     15      #  THEN
     16            Z=0.
     17       ELSE
     18            Z=Y
     19       END IF
     20       ALIMD=Z
     21       RETURN
     22       END
     23 &
     24 &     ---------------------------------
     25 C *
     26 !     SUBMD           PICON
     27 &     *****           *****
     28 !FUN    ALIM,ALIMD;
     29 !STV    Z;
     30 !INP    XP,XM;
     31 !PAR    XMIN,XMAX,PB,TAU,YMIN,YMAX;
     32 !ALG Y;
     33       REAL UP,UM,VP,VM,EPSLON,RANGE
     34 C *
     35 C *   PI-CONTROLLER
     36 C *
     37       RANGE=XMAX-XMIN
     38       UP=(XP-XMIN)/RANGE
     39       UM=(XM-XMIN)/RANGE
     40       VP=ALIM(0.,UP,1.)
     41       VM=ALIM(0.,UM,1.)
     42       EPSLON=(VP-VM)/PB
     43       Z=ALIM(YMIN,Z,YMAX)
     44       Z=ALIMD(EPSLON/TAU,YMIN,Z,YMAX)
     45       Y=ALIM(YMIN,EPSLON+Z,YMAX)
     46       RETURN
     47       END
     48 &
```

Fig. 3.6. A listing of a simple library.

3.4.3. A module

It is of course not possible to meaningfully choose default names for files with a module. The user is advised to choose the names for files with a module to be the name of the module because of the way that the precompiler system produces the names of the derived files. The extension of the source files should be SOU. If all defaults are used then the only information about the files that is needed to be supplied to the precompiler system are the module names.

Precompiler 1 produces two files for each source file with a module. The FORTRAN77 file is named by using the module (!) name and the extension FOR. Precompiler 2 needs the names of the input and output variables and therefore precompiler 1 also produces an information file. Furthermore, in this file the name of the library that was used is placed, if any was used, and the names of the functions and submodules that were used is also placed there. The extension for this file is INF.

Two examples of modules are shown in Fig. 3.7.

```
A:DYSIM\OPB\TNK.SOU                                    02/26/87
------------------------------------------------------------
  1 ! MODUL TNK
  2 ! STV M,TW;
  3 ! INP TI,V1,V2,WO;
  4 ! ALG H;
  5 &
  6       REAL RHO,CP,A,K1,K2
  7       DATA RHO/1000./, CP/4.2E3/, A/1./
  8       DATA K1/1.E7/, K2/100./
  9 &
 10       Q=K1*V1
 11       W1=K2*V2
 12       M : W1-WO
 13       TW: (Q+CP*W1*(TI-TW))/(CP*M)
 14       H=M/(A*RHO)
 15 &
 16       RETURN
 17       END
```

```
A:DYSIM\OPB\REG.SOU                                    02/26/87
------------------------------------------------------------
  1 ! MODUL REG
  2 ! STV Z(2);
  3 ! ALG Y(2);
  4 ! INP M,MSET,TSET,TW;
  5 &
  6 !MACRO PICON
  7 ! STV Z:Z(2);
  8 ! ALG Y:Y(2);
  9 ! INP XP:MSET, XM:M;
 10 ! PAR XMIN=0.8, XMAX=1.2, PB=0.75,
 11 !       TAU=1, YMIN=0, YMAX=1;
 12 !END
 13 &
 14 !MACRO PICON
 15 ! STV Z:Z(1);
 16 ! ALG Y:Y(1);
 17 ! INP XP:TSET, XM:TW;
 18 ! PAR XMIN=40., XMAX=60., PB=0.50,
 19 !       TAU=2, YMIN=0, YMAX=1;
 20 !END
 21 &
 22       RETURN
 23       END
```

Fig. 3.7. A listing of two modules.

3.4.4. The connecting system

The default name for the connecting system is CONSYS and as before the user is advised to use the extension SOU. Precompiler 2 then produces a FORTRAN77 file with the extension FOR. In the source file the precompiler system finds the names of the modules that are being used and on the basis of the module information files a list file is produced. This list file is used by the runtime executive DYSIM86 during a simulation and DYSIM86 assumes that the name is LIST with the extension SEQ so this is the name constructed by the precompiler system. The names of

the submodules and functions that are being used in the modules are recorded in the module information files, and therefore precompiler 2 will also pick these out and place them in a file which is named as the library but with the extension FOR.

An example of a connecting system is given in Fig. 3.8.

```
A:SYSIN\OPS\CONSYS.SOU                                    06/26/87
------------------------------------------------------------------
  1 ! RNR REG, TNK;
  2 ! DOM   MO(2), TI(2), RISET(2), HSET, TSET;
  3 &
  4 ! MOD REG
  5       HSET=HSET.CON
  6       TSET=TSET.CON
  7       M=M.TNK
  8       TI=TI.TNK
  9 ! END
 10 &
 11 ! MOD TNK
 12       Y1=Y1.REG
 13       Y2=Y2.REG
 14       MO=MO1.CON+AMINI(1.,T/RISET1.CON)
 15     &          *(MO2.CON-MO1.CON)
 16       TI=TI1.CON+AMINI(1.,T/RISET2.CON)
 17     &          *(TI2.CON-TI1.CON)
 18 ! END
```

Fig. 3.8. A listing of a connecting system.

3.5. Correction of errors

The precompiler system performs a syntax check and if the syntax is correct it is possible to interpret the source file. If a syntax error is detected in the source file it must be corrected to produce valid output files. The precompiler system is constructed so as to allow correction of errors during precompilation. This interactive control is possible only for certain types of errors.

The precompiler system uses a one pass algorithm which means that each line of the source files is processed exactly one time. The precompiler system also processes each line from left to right. Therefore, only those syntax errors that are caused

by an error on the current line and the current part of the line being processed can be corrected interactively by the precompiler system.

When an error has been detected the line in which it was detected is printed on the screen and the current part of the line that is being processed is indicated by asterixes "*", and this is the part of the line that the user can replace by the correct string of characters. The new string can be shorter or longer but not so long that the new line will exceed 72 characters in length. This limitation is imposed by the FORTRAN77 standard.

In Fig. 3.9 there is an example of first how the erroneous part of a line is indicated by the precompiler system and on the next line how the user can correct the error.

```
Termination character ";" is used as separation character ",".
!ALG  WI;E;
        *

Module : TNK    Line number in programme :        3
You have the following possibilities:
   Replace indicated text with new text in qoutation marks,
   or null input for no corrections,
   or type S for suppression of further error messages.      ','
```

Fig. 3.9. Correction of detected error.

If the user does not want to correct an error a string of blank characters is sent by just pressing the RETURN button. Reasons for not wanting to correct an error can be that the error is caused by a previously detected and uncorrected error or by a previous, undetected error. In this case the standard file editor should be used to correct the error. The precompiler system informs the user about the line number in the source file of the detected (!) error.

If uncorrected or uncorrectable errors are found the output files are not produced and the user must correct the errors before proceeding to the next stage of precompilation.

3.6. Notation for input and output variables

Units (and sections) in a process plant are connected with other units (or sections) via tubes in which matter or information is flowing. Flows can be either entering or leaving the unit and there can be properties and states of substances that are local to the unit.

A mathematical model describing a unit is a series of equations stating the relations among the different flows, properties, etc. using variables for each quantity. These can be split into two categories: the external (i.e. incoming and outgoing flows) and the internal (i.e. local properties) quantities.

A simple example is a heat exchanger. Two flows are streaming through the heat exchanger, one cooling the other. The temperatures and the flow rates of the two flows are examples of external quantities while the temperature of the tube through which heat is transferred from the hot to the cold flow is an example of an internal quantity.

The unit is described by a series of equations expressing some of the internal and external quantities in relation to the other quantities and parameters of the unit, e.g. the heat capacity of the tube in the heat exchanger. Two types of equations are used in a mathematical model of the dynamic behaviour of a unit: differential and algebraic equations. Quantities that are given by a differential equation are called state variables and those given by algebraic equations are called algebraic variables. These are the two types of quantities (internal or external) that can be called output variables. Those external quantities that are not output variables are called input variables.

It is important to notice that input variables may very well be quantities of outgoing flows and that an output variable may be a quantity of an incoming flow. This is illustrated in the following example.

Iamgine a condenser (Fig. 3.10) into which steam flows at a rate W_s. In the condenser the steam pressure is P_s. The steam temperature can be calculated by assuming that the steam is saturated. The resistance to heat transport expressed by the heat transfer coefficient enables the maximum amount of heat transfer from the steam to the tube in which the coolant is flowing to be calculated. Assuming negligible response (acceleration) time the (maximum) steam load W_g can be calculated by an algebraic equation. So this is an example of a quantity of an incoming flow that is an algebraic variable.

The (outgoing) flow of the condensate may be regulated by a valve. The flow rate W_e is related to the valve opening. W_e is an example of a quantity of an outgoing flow that is an input variable.

Now a notation is introduced. External quantities are those related to incoming and outgoing flows and therefore they will be written at the arrows that represent the corresponding flow outside the symbol representing the unit (see P_s, W_g, and W_e as well as W_c, T_{ci}, and T_{co} in Fig. 3.10). W_c is the flow of the coolant and T_{ci} and T_{co} are the temperatures of the incoming and outgoing coolant. Internal quantities are written inside the symbol (see the tube temperature T_t).

Furthermore, those external quantities that are input variables are written to the left of a vertical arrow and above a horizontal arrow (see P_s, W_e, W_c, and T_{ci}). The algebraic and state variables (i.e. the output variables) are written to the right of or below the arrows (see W_g and T_{co}).

Splitting those variables needed (input) in a model from those calculated (output) in the model makes it easy to se whether two models will fit together. Consider the valve introduced

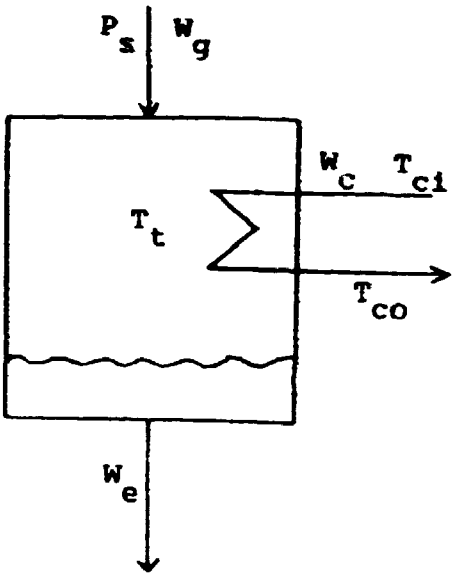


Fig. 3.10. Schematic description of a condenser. Only a few quantities are indicated.

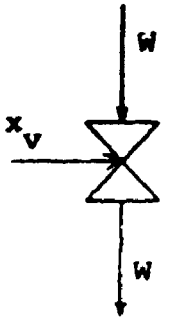


Fig. 3.11. Schematic description of a valve.

above (see Fig. 3.11). The valve opening x_v is an input variable from which the flow W through the valve can be calculated: x_v is above the arrow and W is to the right of the arrows.

The valve is seen to fit to the outgoing flow of the condenser since the flows are written on different sides of the arrows indicating that one of them is an input variable while the other is an output variable. It is also seen that the valve does not fit to the incoming steam flow since they are written on the same sides of the arrows. In this case both are calculated in the model.

4. SIMULATION OF A PART OF A POWER PLANT

4.1. Introduction

The usage of the precompiler system and of the methodology outlined in 3.6 "Notation for input and output variables" is demonstrated with two examples. The first is a dynamic simulation of a part of a power plant. The development of the model is briefly outlined but the details of the model can be obtained elsewhere (KOPOED, 1986).

4.2. Description

A look at a power plant from the outside could be something like that shown in Fig. 4.1. Electricity is produced from a supply of fuel. Furthermore, cold seawater enters and is returned to the sea at an elevated temperature. A look inside the power plant reveals schematically something like that in Fig. 4.2. The part of the power plant that heats and evaporates feedwater can be a nuclear reactor or a boiler. The "model" of this will be seen to be extremely simple. It is included only to close the loop.

The part of the power plant that is really being simulated is the black box with turbine and feedwater preheater chain sections. These are shown schematically in Fig. 4.3. In the figure the mass flows are shown and can be compared with those appearing in Fig. 4.2. The production of electricity by the two turbines is not shown.

The incoming steam passes a steam line and part of the steam flows to the high-pressure (HP) turbine. At two stages steam is extracted by the feedwater preheater chain and the rest flows to a reheater. In the reheater the steam is dried and it is superheated through heat exchange with condensing steam from the steam line. The water that is extracted from the wet steam and the water from the condensing steam are fed to the stream of feedwater in the feedwater preheater chain.

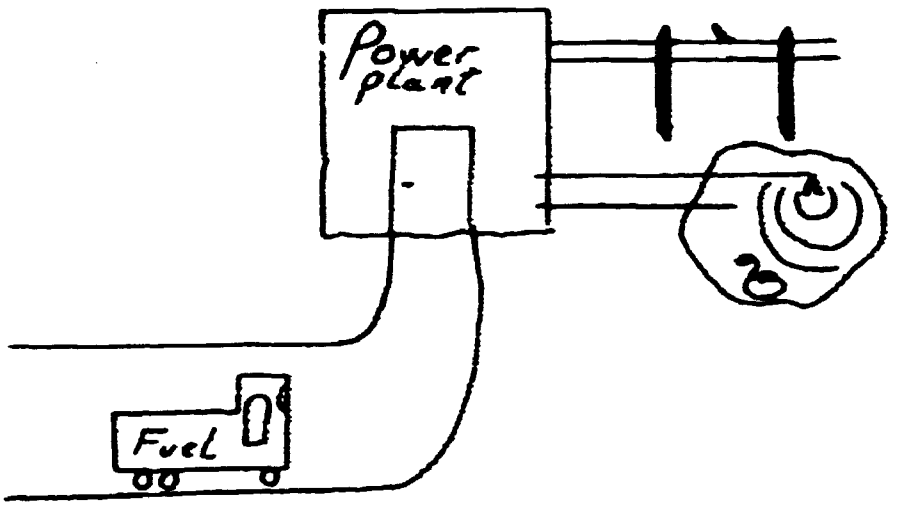


Fig. 4.1. Production of electricity.

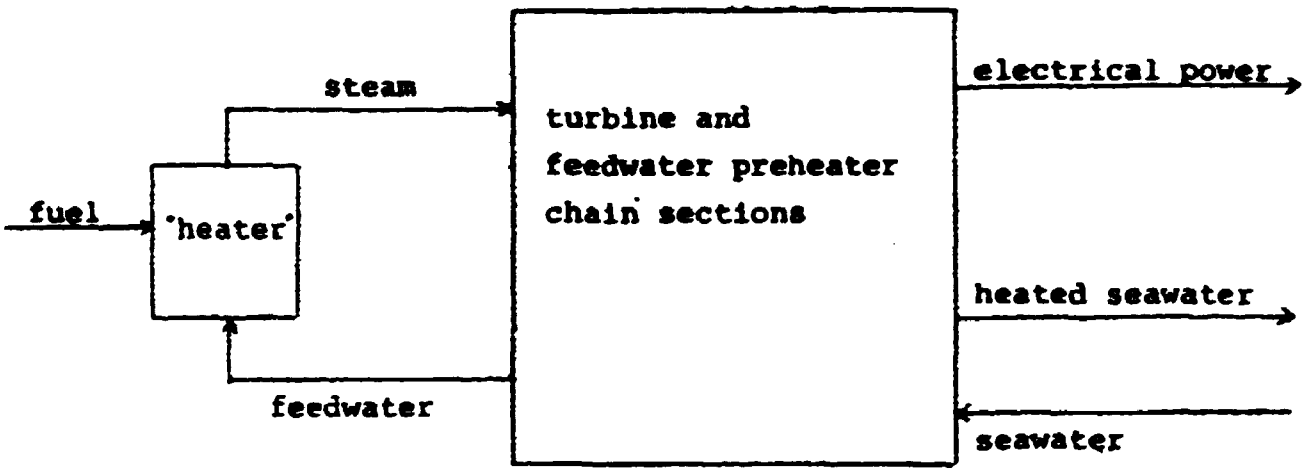


Fig. 4.2. A look inside a power plant.

The superheated steam flows to the low-pressure (LP) turbine where steam is extracted at three stages. The rest of the steam is condensed by cold seawater in the condenser.

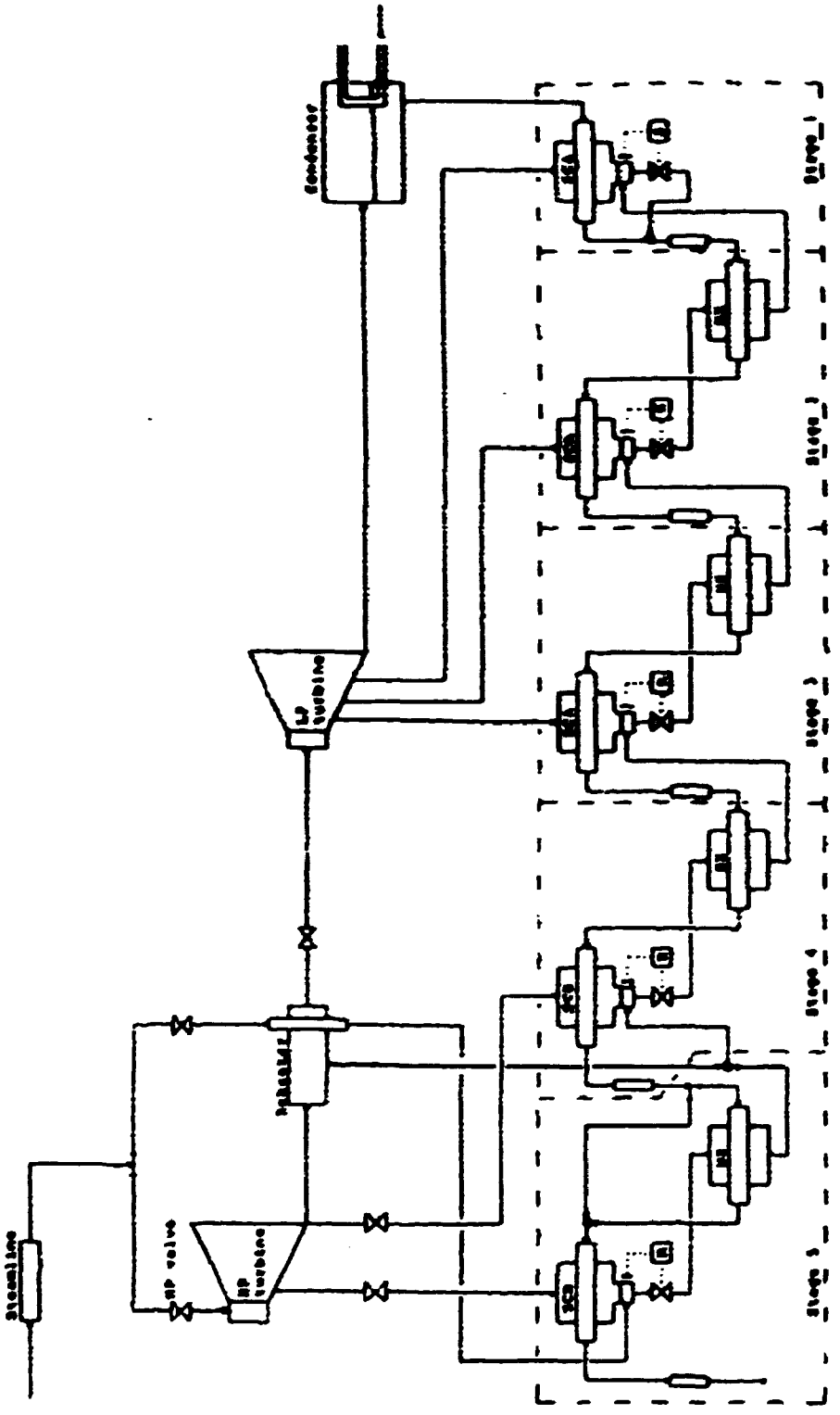


Fig. 4.3. Schematic description of turbine and feedwater preheater chain sections.

In the preheater chain section there are steam coolers with tanks, heat exchangers, PI-controllers and tubes as the most characteristic components. Furthermore, a few valves and mixers and a splitter are present.

The condensate from the condenser in the turbine section flows through the feedwater preheater chain where it passes steam coolers, time delay tubes and heat exchangers whereby it is being heated; it then leaves as feedwater to the boiler. The feedwater is heated in two ways. Steam is being extracted from the turbine section and condenses in the steam coolers through which the feedwater is flowing. Furthermore, this condensate is heating the feedwater in some heat exchangers. The feedwater and the condensate flows countercurrently. In the first steam cooler all the condensate is mixed with the feedwater. The PI-controllers are responsible for controlling the levels of condensate in the steam cooler tanks.

It is seen that the feedwater preheater chain forms a convenient unit for a module, and the other part forms another module (see Fig. 4.4).

The model being used in the simulation will be given in the following sections. Only the equations are given while the derivation is given elsewhere (KOPOED, 1986). Since no submodules are used in the turbine module the model is given here, while the major part of the feedwater preheater chain nodal includes components that are available in the submodule library. The description of the feedwater preheater chain module consists then of an outline of the connections of the components. For the details of the model the reader is referred to Appendix A.

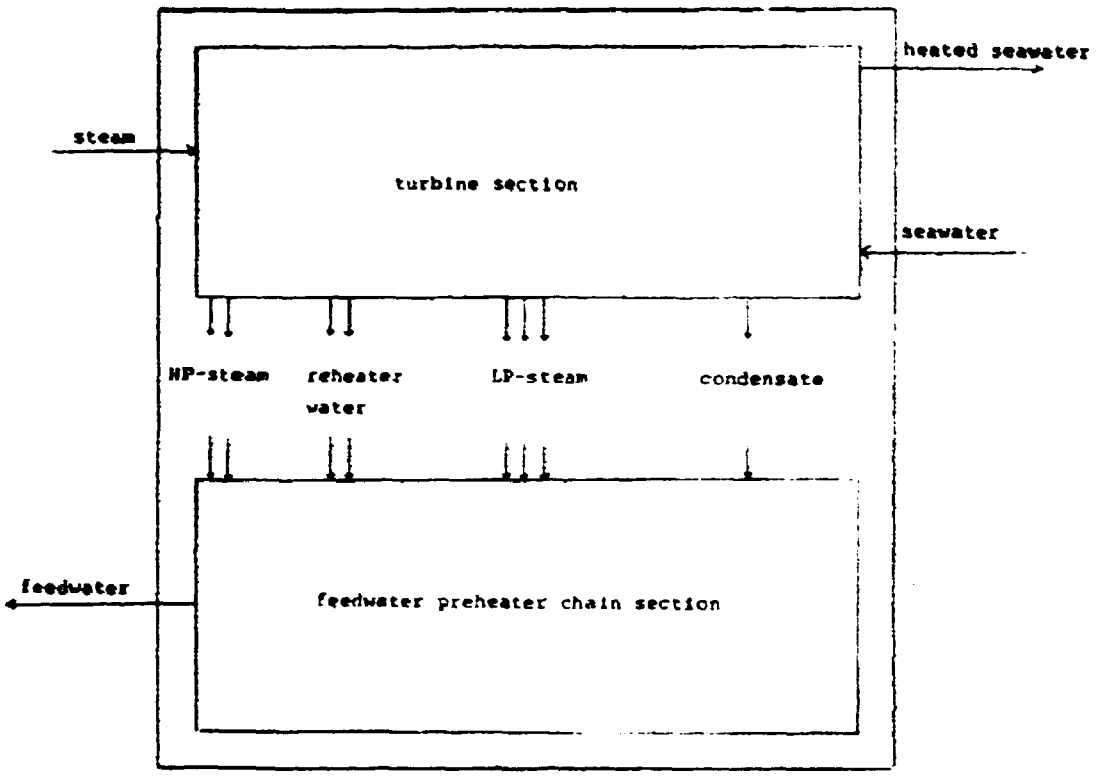


Fig. 4.4. The modular division of Fig. 4.3.

4.3. "Heater model"

It is quite an overstatement to call the following a model of the unit that heats the feedwater and produces steam. Since this project is not concerned with "exact" simulation of process plants, but rather uses simulation for demonstration purposes it was considered to be sufficient to model the turbine and the feedwater preheater chain sections.

The purpose of introducing this "heater model" is merely to close the loop, i.e. make a connection between feedwater and the steam. The steam pressure is controlled ideally, by which is meant that the pressure is constant. It is assumed to be of constant quality, i.e. constant specific enthalpy. The turbine section is allowed to extract just the amount of steam it requires. This can occur if sufficient heating is supplied and by having a large volume of steam. The amount of feedwater to be drawn from the feedwater preheater chain is calculated using a first-order differential equation.

The model then is described by

$$P_e = 70 \text{ bar}$$

$$W_{totin} = (W_{sl} - W_{totin})/\tau \tag{4.1}$$

$$h_v = 0.995 \cdot h_{gs}(P_e) + 0.005 \cdot h_{fs}(P_e) \tag{4.2}$$

where

P_e is the steam pressure
W_{sl} is the flow of steam being extracted from the "heater"
W_{totin} is the flow of feedwater extracted from the preheater chain
h_v is the specific enthalpy of the steam
h_{gs} is a thermodynamic function: specific enthalpy of steam at saturation
h_{fs} is a thermodynamic function: specific enthalpy of water at saturation
τ is a time constant.

In Fig. 4.5 these symbols are written on a schematic presentation of the "heater". Notice that the notation introduced in section 3.6 "Notation for input and output variables" has been used. As noted in the figure input variables are written to the left of vertical arrows and above horizontal arrows. Output variables (state and algebraic variables) are written on the other side of the arrow.

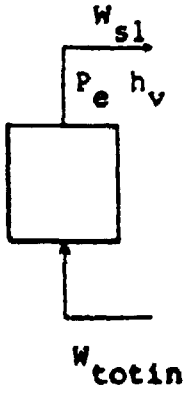


Fig. 4.5. A schematic description of the "heater model".

4.4. Turbine module

4.4.1. Steam line

The steam line is lumped to one chamber by which is meant that it is assumed that the pressure, the specific enthalpy and every other property of the steam is the same at every point in the chamber.

A schematic description of the steam line is given in Fig. 4.6. It is seen to fit the "heater model" since the variables describing the same properties of the steam in the two models are written on opposite sides of the arrows.

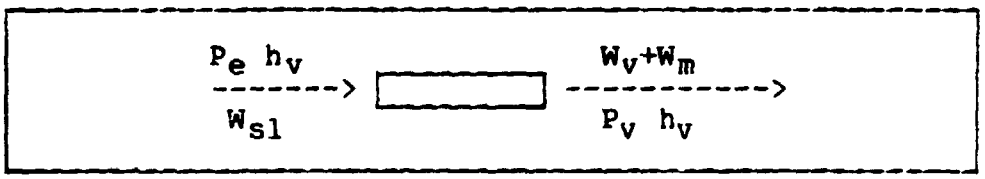


Fig. 4.6. A schematic description of the steam line.

The output variables for the steam line model are W_{sl}, P_v and h_v. The specific enthalpy h_v is assumed to be constant, i.e. no heat loss is taken into account. A quadratic expression describes the relation between steam flow and pressure drop according to Bernoulli's equation (HANSEN and SØLTOFT, 1980, p. 58). The differential equation giving P_v is found by stating conservation of mass.

$$W_{sl} = \sqrt{\frac{P_e - P_v}{k_e}} \tag{4.3}$$

$$\dot{P}_v = \frac{W_{sl} - (W_v + W_m)}{V_{sl}\ \rho' g_v} \tag{4.4}$$

where

k_e is a constant
V_{sl} is the volume of the steam line

$$\rho'_{gv} = \frac{d\rho_{gv}}{dP_v}$$

ρ_{gv} is the density of the steam
W_v is the steam flow to the HP-turbine
W_m is the steam flow to the reheater.

4.4.2. HP-turbine

The steam from the steam line is as already indicated above split into two flows. One of these passes a valve before entering the inlet chamber of the HP-turbine. The steam leaves the inlet chamber and flows through the turbine producing electricity. At two stages steam is extracted from the HP-turbine.

The valve opening determines the flow of steam to the turbines and therefore the power being produced. The flow through the valve is approximated by the expression describing the conditions for a convergent-divergent duct (CHRISTENSEN, 1974) which can be written as

$$W = k \cdot A \cdot P \cdot Y \qquad (4.5)$$

where

k is a constant
A is the area
P is the inlet pressure
Y is a function described below.

The function Y depends on the ratio P_r of the outlet and the inlet pressure. An analytic expression approximating this function adequately well both for ratios below and above the critical ratio 0.577 (CHURCH, 1950, p 74) is given in (CHRISTENSEN, 1979):

$$Y(P_r) = \begin{cases} 1 & \text{for } P_r < 0.577 \\ \sqrt{1-5.5888(P_r-0.577)^2} & \text{for } P_r > 0.577 \end{cases} \qquad (4.6)$$

The flow through the turbine is also described by this expression but another approximation for Y is used. This is because the present simulation is compared with that of an earlier work (CHRISTENSEN, 1986) where the approximation above could not be used because of abnormal conditions.

The specific enthalpy and the steam quality of the steam leaving the HP-turbine is calculated by first assuming 100% efficiency (isentropic expansion) and then correcting for the efficiency measured for the turbine.

A schematic description of the HP-valve and the HP-turbine is shown in Fig. 4.7, and the model becomes as follows:

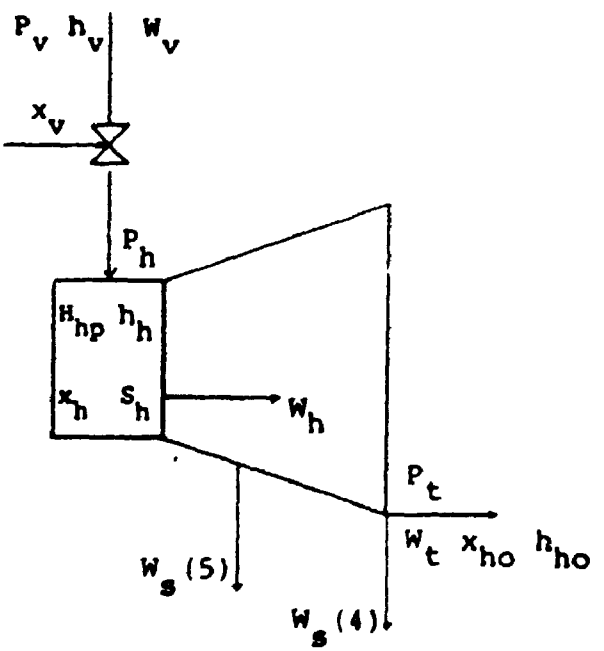


Fig. 4.7. A schematic description of the HP-turbine.

Firstly the steam flow W_v through the HP-valve is calculated

$$W_v = K_v \cdot A_v \cdot P_v \cdot Y_v(P_r) \qquad (4.7)$$

where

$P_r = P_h/P_v$
$Y_v(P_r) = Y(P_r)$, see (4.6)
K_v is a valve constant
A_v is the valve area which is a tabulated function of the valve opening.

Similarly the steam flow W_h out of the inlet chamber is given as

$$W_h = K_h \cdot P_h \cdot Y_h(P_r) \qquad (4.8)$$

where

$P_r = P_t/P_h$
K_h is a constant including the area

$$Y_h(P_r) = \begin{cases} 1 & \text{for } P_r<0.7 \\ 1-(100/9)\cdot(P_r-0.7)^2 & \text{for } P_r>0.7 \end{cases} \qquad (4.9)$$

This approximation for Y_h is also given in CHRISTENSEN, 1979.

Since there is assumed to be no heat loss the specific enthalpy of the steam entering the inlet chamber is the same as it is in the steam line (see (4.2)).

The enthalpy and specific enthalpy of the steam in the inlet chamber is described by

$$\dot{H}_{hp} = W_v \cdot h_v - W_h \cdot h_h \qquad (4.10)$$

$$h_h = \frac{H_{hp}}{V_h \rho' g_h P_h} \qquad (4.11)$$

where

V_h is the volume of the inlet chamber

$$\rho'_{gh} = \frac{d\rho_{gh}}{dP_h}$$

ρ_{gh} is the density of the steam.

Conservation of mass applied to the HP-inlet chamber gives

$$\dot{P}_h = \frac{W_v - W_h}{V_h \, \rho'_{gh}}$$

The specific entropy S_h of the steam in the inlet chamber can be calculated by first calculating the steam quality

$$x_h = \frac{h_h - h_{fs}(P_h)}{h_{gs}(P_h) - h_{fs}(P_h)} \qquad (4.12)$$

$$S_h = x_h \cdot S_{gs}(P_h) + (1 - x_h) \cdot S_{fs}(P_h) \qquad (4.13)$$

where

S_{gs} is the specific entropy of steam at saturation
S_{fs} is the specific entropy of water at saturation.

At the outlet of the HP-turbine the pressure is P_t and if it is assumed that the turbine efficiency is 100% the entropy is not changed, i.e. the process of steam expansion is isentropic. In this case the steam quality and specific enthalpy can be calculated as

$$x'_{ho} = \frac{S_h - S_{fs}(P_t)}{S_{gs}(P_t) - S_{fs}(P_t)} \qquad \text{(100\% efficiency)} \qquad (4.14)$$

$$h'_{ho} = x'_{ho} h_{gs}(P_t) + (1 - x'_{ho}) h_{fs}(P_t) \qquad \text{(100\% efficiency)} \qquad (4.15)$$

Assuming 100% efficiency the enthalpy drop is $(h_h - h'_{ho})$. Now the efficiency is γ_h and therefore the enthalpy drop is $\gamma_h(h_h - h'_{ho})$, (SMITH and VAN NESS, 1975), and the steam quality and specific entropy can be calculated at the HP-turbine outlet.

$$h_{ho} = h_h - \gamma_h(h_h - h'_{ho}) \qquad \text{(efficiency } \gamma_h\text{)} \qquad (4.16)$$

$$x_{ho} = \frac{h_{ho} - h_{fs}(P_t)}{h_{gs}(P_t) - h_{fs}(P_t)} \qquad \text{(efficiency } \gamma_h\text{)} \qquad (4.17)$$

γ_h is dependent on the working conditions including the steam load, but in this model it is kept constant. The value is for typical working conditions.

The feedwater preheater chain extracts steam (steam load) from the HP-turbine at two stages. The flows are $W_s(4)$ and $W_s(5)$ and mass balance yields for the steam flow W_t out of the turbine

$$W_t = W_h - (W_s(4) + W_s(5)) \qquad (4.18)$$

This is valid in steady state but by assuming negligible response time this is also used in transients.

At two stages steam is being extracted. Later the pressure and the specific enthalpy of this steam as well as the power being produced will be calculated.

4.4.3. Reheater

The main functions of the reheater is to dry the steam from the HP-turbine and to superheat it before entering the LP-turbine. Due to this the reheater is highly asymmetric and it is therefore split into two compartments (see Fig. 4.8) the moisture separator and the superheater. In the moisture separator the steam is dried before entering the superheater. The water that is removed in the moisture separator flows to the feedwater preheater chain. The steam is being superheated by condensing steam from the steam line in the superheater. This condensate also flows to the feedwater preheater chain.

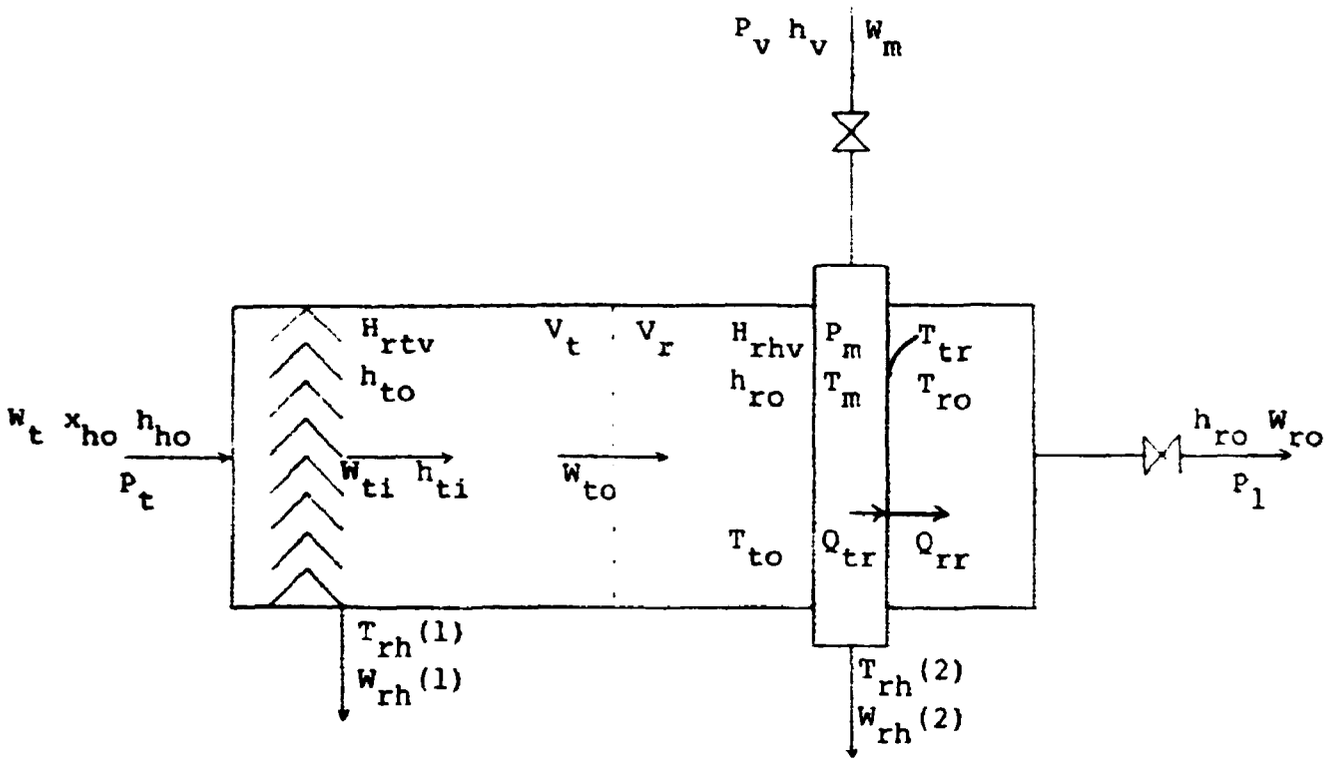


Fig. 4.8. A schematic description of the reheater.

It is assumed that the steam quality is raised to 99.5% in the moisture separator so the flow of dried steam and the specific enthalpy become

$$W_{ti} = \frac{x_{ho}}{0.995} W_t \tag{4.19}$$

$$h_{ti} = 0.995 \cdot h_{gs}(P_t) + 0.005 \cdot h_{fs}(P_t) \tag{4.20}$$

In the other end the steam leaves through a valve and an expression like (4.5) should be used, but since the pressure drop is small this is again replaced by another expression. Since the range of the pressure drop $(P_t - P_l)$ is small a linear relationship is used

$$W_{ro} = K_{hl} \cdot A_{ltv} \cdot (P_t - P_l) \tag{4.21}$$

where

K_{hl} is a constant including the area
A_{ltv} is valve opening (range: 0-1).

Assuming no pressure gradients in the reheater application of conservation of mass yields

$$\dot{P}_t = \frac{W_{ti}-W_{ro}}{(V_t+V_r)\rho'_{gr}} \tag{4.22}$$

where

V_t+V_r is the volume of the reheater

$$\rho'_{gr} = \frac{d\rho_{gr}}{dP_t}$$

ρ_{gr} is the density of the steam.

The assumption of no pressure gradients means that the flow W_{to} of steam from the moisture separator to the superheater must be

$$W_{to} = \frac{V_r \cdot W_{ti}+V_t \cdot W_{ro}}{V_t+V_r} \tag{4.23}$$

The enthalpy H_{rtv} and the specific enthalpy h_{to} of the steam in the moisture separator is described by

$$\dot{H}_{rtv} = W_{ti} \cdot h_{ti}-W_{to} \cdot h_{to} \tag{4.24}$$

$$h_{to} = \frac{H_{rtv}}{V_t \ \rho'_{gr} \ P_t} \tag{4.25}$$

and the specific enthalpy h_{ro} of the steam in the superheater is

$$h_{ro} = \frac{H_{rhv}}{V_r \ \rho'_{gr} \ P_t} \tag{4.26}$$

where H_{rhv} is the (total) enthalpy of the steam in the superheater.

The heat transfer in the superheater is calculated in a way similar to that for a steam cooler which is described in Appendix A. The heat transfer coefficient on the secondary side is assumed to depend linearly on the steam flow

$$k_{qr} = \frac{W_{ro}}{C_{kqr}} \tag{4.27}$$

where C_{kqr} is a constant.

The total heat transfer coefficient kL is determined from

$$\frac{1}{kL} = \frac{1}{k_{qr}} + \frac{1}{k_{qt}} \tag{4.28}$$

where k_{qt} is the heat transfer coefficient on the primary side (a constant).

The auxiliary variable

$$z = \frac{kL}{W_{ro} \cdot c_{pr}} \tag{4.29}$$

where

c_{pr} is the specific heat capacity of the steam

is calculated and the weight factor

$$B = \frac{1}{z} - \frac{\exp(-z)}{1-\exp(-z)} \tag{4.30}$$

is used in the calculation of the mean temperature

$$T_{rm} = B \cdot T_{to} + (1-B) \cdot T_{ro} \tag{4.31}$$

of the steam on the secondary side. The steam entering the

superheater is assumed to be saturated so the temperature can be found from a thermodynamic function

$$T_{to} = T_{sat}(P_t) \tag{4.32}$$

The temperature of the steam at the outlet is determined by the degree of superheating

$$T_{ro} = T_{to} + \frac{h_{ro}-h_{gs}(P_t)}{c_{pr}} \tag{4.33}$$

The tube is now introduced in the calculations. The heat flow to the tube from the condensing steam on the primary side and the heat flow from the tube to the secondary side is

$$Q_{tr} = k_{qt}\cdot(T_m-T_{tr}) \tag{4.34}$$
$$Q_{rr} = k_{qr}\cdot(T_{tr}-T_{rm}) \tag{4.35}$$

where

$$T_m = T_{sat}(P_m) \tag{4.36}$$

is the temperature of the condensing steam.

Conservation of energy on the secondary side now gives an expression with the enthalpy of the steam in the superheater

$$\dot{H}_{rhv} = W_{to}\cdot h_{to} + Q_{rr} - W_{ro}\cdot h_{ro} \tag{4.37}$$

and when applied to the tube

$$\dot{T}_{tr} = \frac{Q_{tr}-Q_{rr}}{C_{tr}} \tag{4.38}$$

where C_{tr} is the heat capacity of the tube.

Applying energy conservation on the primary side gives the amount of steam that is condensing to account for the heat flow

$$W_{mc} = \frac{Q_{tr}}{h_v - h_m} \tag{4.39}$$

where

$$h_m = h_{fs}(P_m) \tag{4.40}$$

is the specific enthalpy of the water.

This is the flow of steam leaving the inside of the tube on the primary side. The flow of steam entering through a valve is

$$W_m = K_{vm} \cdot A_{mov} \cdot (P_v - P_m) \tag{4.41}$$

where

K_{vm} is a constant including the area
A_{mov} is the valve opening

see (4.21). Application of mass conservation leads to

$$\dot{P}_m = \frac{\dot{W}_m - \dot{W}_{mc}}{V_{mo}\ \rho'_{gh}} \tag{4.42}$$

where

V_{mo} is the volume of steam inside the tube

$$\rho'_{gh} = \frac{d\rho_{gh}}{dP_m}$$

ρ_{gh} is the density of the steam.

The flows and the temperatures of the water leaving the reheater are

$$W_{rh}(1) = W_t - W_{ti} \quad (4.43)$$

$$W_{rh}(2) = W_{mc} \quad (4.44)$$

$$T_{rh}(1) = T_{to} \quad (4.45)$$

$$T_{rh}(2) = T_m \quad (4.46)$$

4.4.4. LP-turbine

The LP-turbine is very much like the HP-turbine. The steam in the inlet chamber is superheated however, so a thermodynamic function S_{gsu} giving the extra contribution to the specific entropy is used. A schematic description of the LP-turbine is shown in Fig. 4.9.

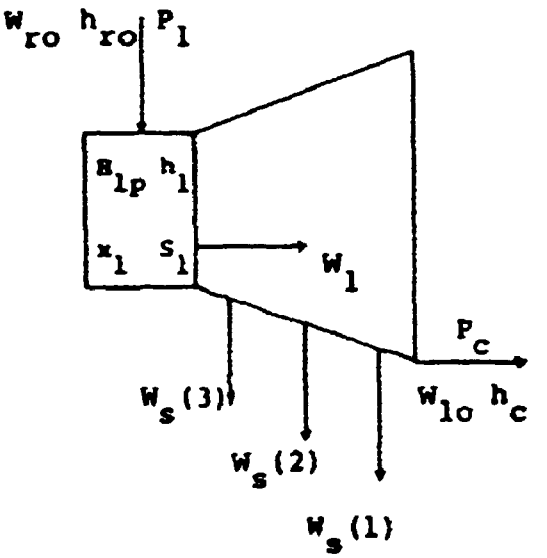


Fig. 4.9. A schematic description of the LP-turbine.

The equations are given here in order to make the model description complete.

$$W_l = K_l \cdot P_l \cdot Y_l(P_r) \quad (4.47)$$

where

$P_r = P_c/P_1$

P_c is the steam pressure in the condenser

$$Y_1(P_r) = Y_h(P_r) \tag{4.48}$$

$$\dot{P}_1 = \frac{W_{ro}-W_1}{V_1\ \rho'_{g1}} \tag{4.49}$$

where

V_1 is the volume of the steam in the inlet chamber

$$\rho'_{g1} = \frac{d\rho_{g1}}{dP_1}$$

ρ_{g1} is the density of the steam

$$h_1 = \frac{H_{1p}}{V_1\ \rho'_{g1}\ P_1} \tag{4.50}$$

$$\dot{H}_{1p} = W_{ro}\cdot h_{ro} - W_1\cdot h_1 \tag{4.51}$$

$$S_1 = S_{gs}(P_1) + S_{gsu}(P_1, h_1-h_{gs}(P_1)) \tag{4.52}$$

$$x'_c = \frac{S_1-S_{fs}(P_c)}{S_{gs}(P_c)-S_{fs}(P_c)} \qquad \text{(100\% efficiency)} \tag{4.53}$$

$$h'_c = x'_c\cdot h_{gs}(P_c)+(1-x'_c)\cdot h_{fs}(P_c) \qquad \text{(100\% efficiency)} \tag{4.54}$$

$$h_c = h_1-\gamma_1(h_1-h'_c) \qquad \text{(efficiency } \gamma_1\text{)} \tag{4.55}$$

$$W_{1o} = W_1-(W_s(1)+W_s(2)+W_s(3)) \tag{4.56}$$

4.4.5. Condenser

In the condenser the wet steam from the LP-turbine is condensed and the heat is removed by a flow of seawater. The condensate is flowing to the feedwater preheater chain. An extremely simple

model is applied. A schematic description of the condenser is given in Fig. 4.10.

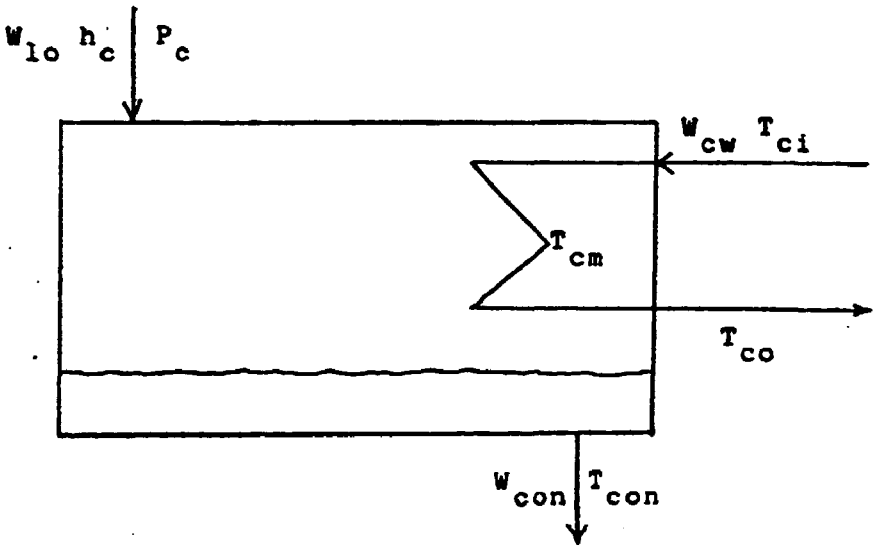


Fig. 4.10. A schematic description of the condenser.

The temperature T_{con} of the condensate is taken to be the same as the temperature of the tube. It is assumed that the flow of seawater is constant so that a constant weight factor B_c can be applied. The temperature has been used instead of enthalpy as the state variable in the equations derived from application of energy conservation. It is assumed that all the incoming steam is condensing.

The model is described by the following equations.

$$P_c = P_{sat}(T_{con}) \qquad (4.57)$$

where P_{sat} is a thermodynamic equation giving the steam pressure at saturation

$$h_{fsc} = h_{fs}(P_c) \tag{4.58}$$

$$Q_{ct} = W_{lo}(h_c - h_{fsc}) \tag{4.59}$$

$$T_{cm} = B_c \cdot T_{ci} + (1-B_c) \cdot T_{co} \tag{4.60}$$

$$Q_{cw} = k_{cw}(T_{con} - T_{cm}) \tag{4.61}$$

where k_{cw} is the heat transfer coefficient on the inside of the tube (a constant).

Let C_{ct} and C_{cw} be the heat capacities of the tube and of the sea water inside the tube and apply energy conservation to get

$$\dot{T}_{con} = \frac{Q_{ct} - Q_{cw}}{C_{ct}} \tag{4.62}$$

$$\dot{T}_{co} = \frac{Q_{cw} + W_{cw}(T_{ci} - T_{co})c_{pc}}{C_{cw}} \tag{4.63}$$

where c_{pc} is the specific heat capacity of the sea water.

It is assumed that there is always enough condensate in the condenser when the feedwater preheater chain is extracting feedwater, but a state variable is introduced to gauge the amount of condensate that is being accumulated.

$$\dot{M}_{ac} = W_{lo} - W_{con} \tag{4.64}$$

where W_{con} is the flow of feedwater extracted by the chain.

4.4.6. Steam extraction

At five stages at the two turbines (two on the HP- and three on the LP-turbine) steam is being extracted at rates calculated in the feedwater preheater chain module. From static data the following expressions were found to describe the pressures and the specific enthalpy at the five stages:

$$P_s(i) = P_c + p_{fvr}(i)(P_l - P_c) \qquad i=1,2,3 \tag{4.64a-c}$$

$$P_s(4) = P_t \tag{4.64d}$$

$$P_s(5) = P_t+(a_1+a_2 \cdot W_h)(P_h-P_t) \qquad (4.64e)$$

$$h_{si}(i) = h_c+h_{fvr}(i)(h_1-h_c) \qquad i=1,2,3 \qquad (4.65a\text{-}c)$$

$$h_{si}(4) = h_{ho} \qquad (4.65d)$$

$$h_{si}(5) = h_{ho}+h_{fvr5}(h_h-h_{ho}) \qquad (4.65e)$$

where the constants

$p_{fvr}(i)$ $\quad i=1,2,3$

$h_{fvr}(i)$ $\quad i=1,2,3$

a_1 and a_2

h_{fvr5}

are determined from static data.

4.4.7. Power calculation

The power being produced by the two turbines is calculated by assuming that there is no loss of heat. Therefore the power being produced is the drop in enthalpy

$$E_h = W_h(h_h-h_{ho}) - W_s(5)(h_{si}(5)-h_{ho}) \qquad (4.66)$$

$$E_l = W_l(h_l-h_c) - \sum_{i=1}^{3} W_s(i)(h_{si}(i)-h_c) \qquad (4.67)$$

4.5. Feedwater preheater chain

The feedwater to the boiler is heated by poor quality steam extracted from the two turbines. The steam is condensed at several places:

1) poor-quality steam in the steam coolers in the preheater chain;

2) high-quality steam on the primary side in the reheater in the turbine section

3) in the HP-turbine, when drying the steam in the reheater the water is extracted; and finally

4) in the condenser.

All these flows can be seen to mix after the first stage of five in the preheater chain (see Fig. 4.3).

The chain can be split into five stages of three different kinds. The three stages in the middle are one kind while the two in the ends are different. The first stage (with the coldest feedwater) consists only of a steam cooler while the other four also include heat exchangers. In stage 5 some of the feedwater bypasses the heat exchanger. In connection with every steam cooler there is a PI-regulation of the water level in the tank (see the description of steam coolers, using a valve determining the flow of condensate).

Furthermore, flows are being mixed in tubes or tanks and being delayed in tubes.

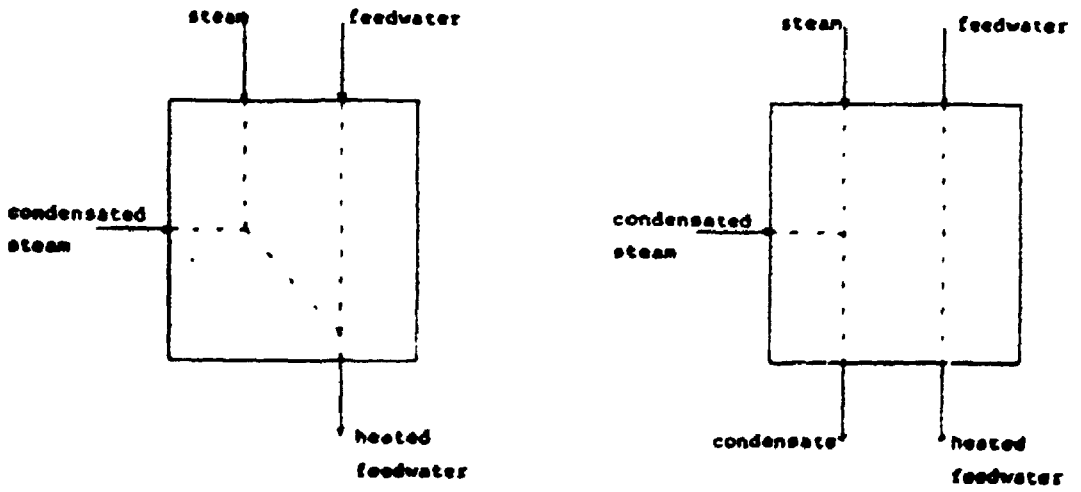


Fig. 4.11. Schematic description of the stages in the feedwater preheater chain.

In each stage steam extracted from the turbines heats the feedwater (see Fig. 4.11). The steam condensates and is mixed with a flow of steam condensated elsewhere. In stages 2-5 this "mixture" of condensated steam is leaving the stage as a flow of condensate while in stage 1 the condensate is mixed with the feedwater before leaving the stage. This is the point where all the condensate is mixed and from this stage on is renamed "feedwater" (see Fig. 4.12). The term co. symbolizes the condensate extracted from the condenser and h.fw. is the heated feedwater. S1, S2, ..., S5 are the steam loads of the turbines and C1 and C2 is water from the reheater (see Fig. 4.3).

The steam flows W_s are calculated here in the feedwater preheater model while the pressures P_s and the specific enthalpies h_{si} are calculated in the turbine model. The temperatures T_{rh} and the flows W_{rh} of water from the reheater are calculated in the turbine module as well as the temperature T_{fi} of the condensate from the condenser while the condensate flow W_{fi} is calculated in this module. In the turbine model T_{fi} and W_{fi} are termed T_{con} and W_{con}.

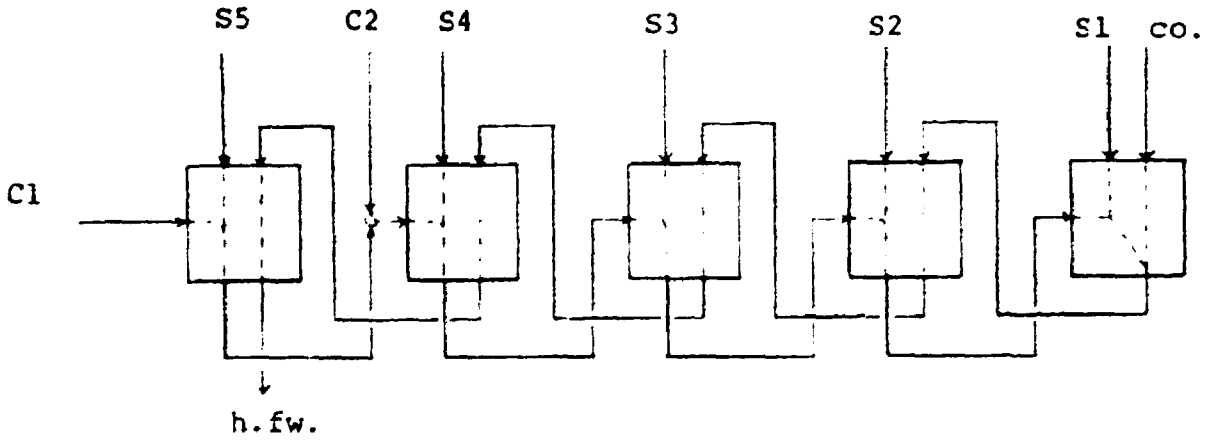


Fig. 4.12. Interconnection of the five stages.

4.5.1. Stage 1

The feedwater extracted from the condenser is heated by condensing steam which in a tank is mixed with a flow of condensate from stage 2. This is described by a submodule, a steam cooler. A PI-controller operates a valve that determines the flow of condensate leaving the tank. The controller will try to maintain a given water level in the tank. This description of the steam cooler, the valve and the PI-controller is similar to that of the other stages.

The flow of condensate, W_{fi}, is mixed with a flow, W_{do}, of condensate from the tank of the steam cooler to a total flow W_{totin} of feedwater. This flow is calculated in the reactor model and since the PI-controller determined the flow W_{do} of condensate the flow rate W_{fi} can be calculated

$$W_{fi} = W_{totin} - W_{do} \tag{4.68}$$

Stage 1 is indicated in Fig. 4.3. For the time delay see next subsection.

4.5.2. Stages 2-4

In stages 2-4 the feedwater is heated in a heat exchanger before it is heated in the steam cooler. All the connecting tubes between stages are gathered in lumps for time delays.

The delays are inserted in front of the heat exchanger. The steam cooler, the valve and the PI-controller are described in 4.5.1.

Condensate leaving the steam coolers is used to heat feedwater in the heat exchangers. In this way the condensate is flowing through the stages in the opposite direction of the feedwater.

Stages 2, 3 and 4 are indicated in Fig. 4.3.

4.5.3. Stage 5

Since stage 5 is the last stage the condensate entering the tank cannot stem from another stage. Instead the condensed steam from the primary side of the reheater is used. With this minor difference the descriptions of the steam cooler, the valve and the PI-controller are as for stage 1.

Some of the feedwater entering the stage bypasses the heat exchanger. This is done by splitting the flows and re-mixing the feedwater before entering the steam cooler.

$$W_{5,1} = SPLIT \cdot W_m(1) \tag{4.69}$$

$$W_{5,2} = (1-SPLIT) \cdot W_m(1) \tag{4.70}$$

where $W_m(1)$ is flow being split.

The final different feature is that the condensate leaving the heat exchanger is mixed with the water removed from the wet steam in the reheater prior to be passed on to stage 4 (see Fig. 4.12 and Fig. 4.3).

There is a tube in stage 5 and therefore a delay (see Fig. 4.3).

4.5.4. Valves

Stages 1-3 are connected to the LP-turbine while the steam to stages 4 and 5 comes from the HP-turbine. Stages 1-3 are connected directly via tubes to the LP-turbine and the steam coolers determine the steam load on the LP-turbine. On the tubes connecting stages 4 and 5 with the HP-turbine there are valves, one in each tube. These valves determine the steam load on the HP-turbine. These valves are check valves (non-return or -reflux) and the flow through the valves are approximated by a linear relationship with the pressure drop,

$$W_{fv}(i) = \left\{ \begin{array}{ll} 0 & \text{if } P_s(i) < P_{fv}(i) \\ 400(P_s(i)-P_{fv}(i)) & \text{if } P_s(i) > P_{fv}(i) \end{array} \right\} i=4,5 \tag{4.71}$$

where $P_s(i)$ is the extraction pressure (4.64d-e), and $P_{fv}(i)$ is the pressure in the steam cooler. The valves are not controlled in this model.

It is seen that two types of steam coolers are needed: type A used in stages 1-3 and type B for 4 and 5. In type A the steam load is calculated in the steam cooler and the pressure is the same as the extraction pressure from the HP-turbine (4.64a-c). In type B the inlet steam flow is the flow through the valve and the pressure in the steam cooler is calculated using the conservation of mass relationship.

PI-controllers control the drainage level in the steam cooler tanks. A linear model is used

$$W_{do}(i) = c_v(i)\ Y(i) \qquad i=1,2,\ldots,5 \qquad (4.72)$$

where Y(i) is the PI-controller output (unit range) and $c_v(i)$ is the valve coefficient.

4.5.5. The chain

The components of the feedwater preheater chain are connected with each other as presented schematically in Fig. 4.3. The components are:

1) steam cooler, types A and B (submodules)
2) heat exchanger (submodule)
3) PI-controller (submodule)
4) mixer (submodule)
5) time delay tube (submodule)
6) valve
7) flow splitter

As indicated all but the valves and the flow splitter are modelled in the library as submodules. The only executable Fortran statements are then

1) calculation of the flows through valves (one statement/ valve) (see notes 3 and 5 in Table 4.1 and Eqs. (4.71-72));

2) calculation of the flows in the splitter (two statements) (see notes 1 and 2 in Table 4.1 and Eqs. (4.69-70));

3) calculation of the condensate flow from the condenser (one statement) (see Eq. (4.68)); and

4) conversion from mass flow to volumetric flow (see Eq. (4.73) below).

The time delay submodules operate with volumetric flows, so for each time delay there is a (single) statement converting mass flow into volumetric flow

$$V(i) = W(i)/\rho(i) \tag{4.73}$$

where W(i) is the mass flow and $\rho(i)$ is the density of the feedwater

$$W(i) = \begin{cases} W_m(1) & \text{for } i=1,2,3,4 \\ W_m(3) & \text{for } i=5 \end{cases} \tag{4.74}$$

where $W_m(1)$ is the flow out of the mixer in stage 1 (W_{totin} in (4.68)), and $W_m(3)$ (also W_{totin}) is the flow out of the feedwater mixer in stage 5 (not the drainage mixer).

The rest of the components are modelled in submodules and the connection between components are established by transferring an output variable from one component to an input variable of the corresponding component connected to the first (see Table 4.1).

As an example, let us consider the flow of feedwater from the heat exchanger to the steam cooler in stage 2. The feedwater flow $W_m(1)$ is calculated in stage 1 as the total flow from a mixer. In Table 4.1 the flow of feedwater in the heat exchanger,

		Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Heat exchanger	W_c	-	WH(1)	WH(1)	WH(1)	WS1[1]
	T_{ci}	-	TOD(1)	TOD(2)	TOD(3)	TOD(4)
	T_{co}	-	TOH(1)	TOH(2)	TOH(3)	TOH(4)
	W_w	-	WDO(2)	WDO(3)	WDO(4)	WDO(5)
	T_{wi}	-	TD(4)	TD(3)	TD(4)	TD(5)
	T_w	-	TW(1)	TW(2)	TW(3)	TW(4)
Steam coolers	P_s	PS(1)	PS(2)	PS(3)	PFV4	PFV5
	W_s	WS(1)	WS(2)	WS(3)	WFV4[3]	WFV5[3]
	h_{si}	HSI(1)	HSI(2)	HSI(3)	HSI(4)	HSI(5)
	h_{sb}	HSB(1)	HSB(2)	HSB(3)	dummy[4]	dummy[4]
	W_f	WF1	WH(1)	WH(1)	WH(1)	WH(3)
	T_{fi}	TF1	TOH(1)	TOH(2)	TOH(3)	TH(3)
	T_o	TOS(1)	TOS(2)	TOS(3)	TOS(4)	TOS(5)
	W_d	WDO(2)	WDC(3)	WDO(4)	WH(2)	WRH(2)
	T_{di}	TW(1)	TW(2)	TW(3)	TW(2)	TRH(2)
	$-W_{da}$[5]	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)
	T_d	TD(1)	TD(2)	TD(3)	TD(4)	TD(5)
	H	H(1)	H(2)	H(3)	H(4)	H(5)
Mixer	W_1	WDO(1)			WDO(5)	WS1[1]
	T_1	TD(1)			TW(4)	TOH(4)
	W_2	WF1			WRH(1)	WS2[2]
	T_2	TOS(1)			TRH(1)	TOD(4)
	W_o	WH(1)			WH(2)	WH(3)
	T	TW(1)			TW(2)	TW(3)
Si-regulator	x	H(1)	H(2)	H(3)	H(4)	H(5)
	x_o	HSET(1)	HSET(2)	HSET(3)	HSET(4)	HSET(5)
	y	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)
Time-delay	$-W_w$[6]	WH(1)	WH(1)	WH(1)	WH(1)	WH(3)
	T_i	TW(1)	TOS(2)	TOS(3)	TOS(4)	TOS(5)
	T_o	TOD(1)	TOD(2)	TOD(3)	TOD(4)	TOD(5)

1) see (4.71)
2) see (4.72)
3) see (4.69)
4) not used, non-reflux valve, see (4.73)
5) see (4.70)
6) see (4.74)

Table 4.1. Connections between components in the feedwater preheater chain.

W_c, and in the steam cooler, W_f, are both $W_m(1)$ which is represented by the Fortran variable WM(1). In the heat exchanger the outlet temperature is T_o, represented by TOH(1). The feedwater inlet temperature in the steam cooler is T_{fi} and this is seen to be TOH(1) as well. So the connection between the steam cooler and the heat exchanger is established correctly as far as the feedwater, stage 2, is concerned.

Because a sorting facility is not (yet) available, the Fortran statements and the macros including submodules must be written in a sequence as close as possible to the "direction of flows" in the model.

4.6. Parameters

Now that the model has been developed the next steps are acquisition of data and making a computer programme. As was described earlier (3.3.4 "Connecting commands") the data can be categorized as one of two types. One type that are going to be changed between simulations and a type that is constant.

The data for this simulation are listed in Appendix K. The major part of the data are of the second (constant) type and therefore they are made an integrated part of the model and will be written in the modules.

4.7. Simulation

A preliminary version of the precompiler system was used to make the FORTRAN77 programme. Listings of the source files to the precompiler system are found in Appendix C. It can be seen that the syntax for the connecting system does not follow the description in Chapter 3 but this is because the syntax has been changed. A few of the features in the precompiler system have been demonstrated and therefore a few unnecessary commands and statements appear. The constructions concerning writing reports on a file appearing after the OUT command in the connect-

ing system are quite unreliable and should not be used in other programmes. (For further information see the report (KOFOED, 1986) that describes this model in detail.)

The steps to go through when a simulation must be performed are:

1) find the steady state,
2) determine the maximum step size, and
3) introduce disturbances and run the transients.

These steps can be performed by merely changing the input file. Both the input file and the steps above are described in the report (KOFOED, 1986).

Here the results of the simulation will be described briefly. The steady state that was determined was as close to that of an earlier study of the same power plant as could be expected due to the extra simplifications that were made in this simulation. The Jacoby matrix calculated in step two led to the conclusion that with a second-order Runge-Kutta, which was used as the integration routine in all simulation experiments, the maximum time step is 0.06 s.

Three transients were simulated: 10% increase and 10% decrease of the HP-valve opening and 50% decrease in seawater flow through the condenser. The last transient is "illegal" since it was explicitly assumed that the seawater flow is constant but the inaccuracy that is introduced by ignoring this is small. The total power ET and the temperature TOD5 of feedwater from the preheater chain are plotted for each of the three transients.

The valve position is changed by a linear ramp in the connecting system. It is seen (Fig. 4.13) that the system is nonlinear since the relative changes in the total power in the two cases are not the same. It is also seen that the temperature of the feedwater does not change instantly. This is because of the time delay from the fifth steam cooler to the heater (see Fig. 4.3). If the temperature at the outlet of the steam cooler had been plotted it would have shown an "instant" change.

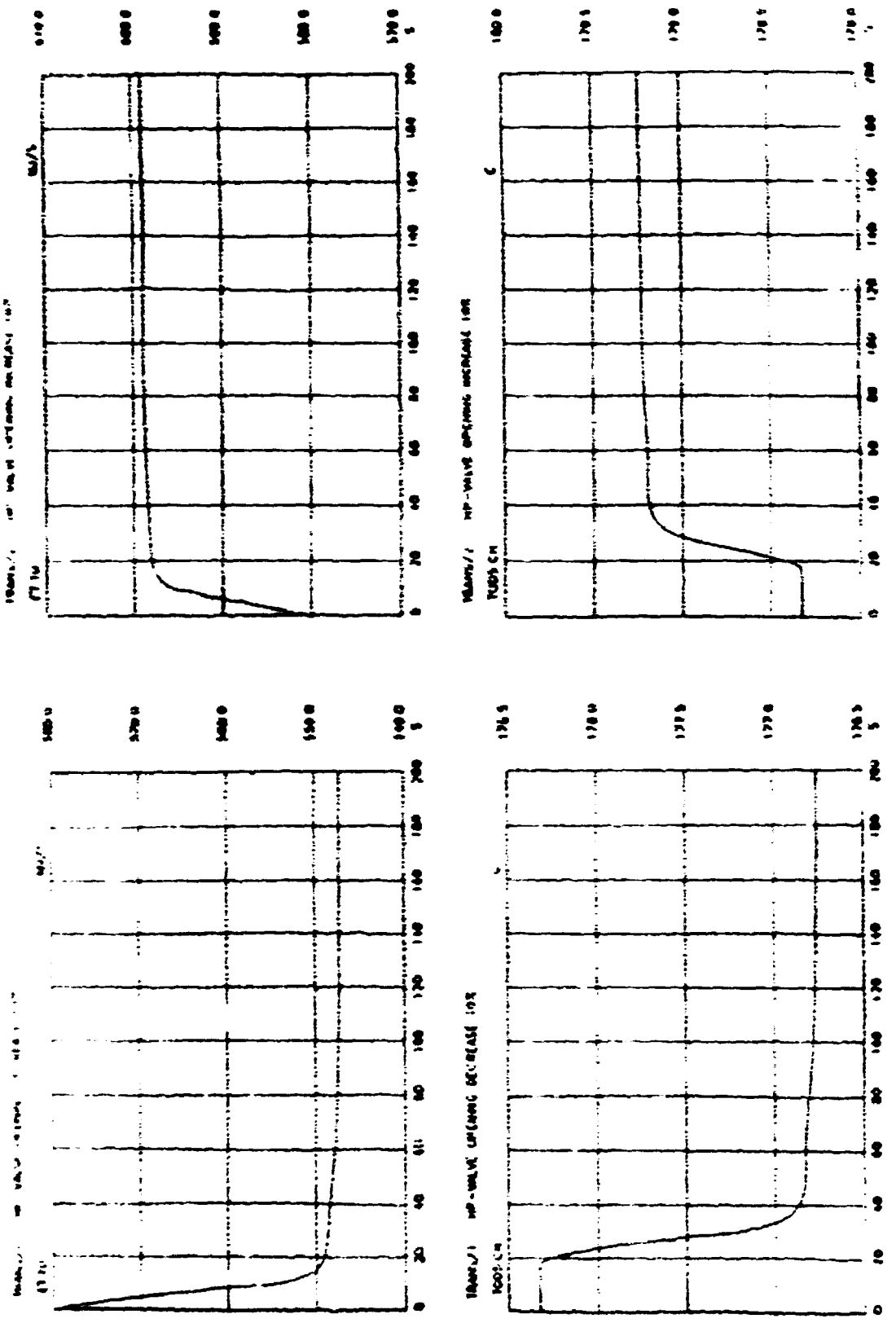


Fig. 4.13. Plots of the first two transients: changes of HP-valve opening.

The third transient can be thought of as being caused by the failure of one out of two parallel pumps. It is seen (Fig. 4.14) that this gives a decrease in the power that is being produced but there is almost no change in the temperature of the feed-water. The reduced flow of coolant increases the temperature and the pressure in the condenser, and therefore the decrease in steam enthalpy across the turbines becomes smaller resulting in a drop in turbine power. The feedwater chain acts in such a way that although the temperature of condensate in the condenser is increased then the temperature of the feedwater entering the heater is not affected significantly.

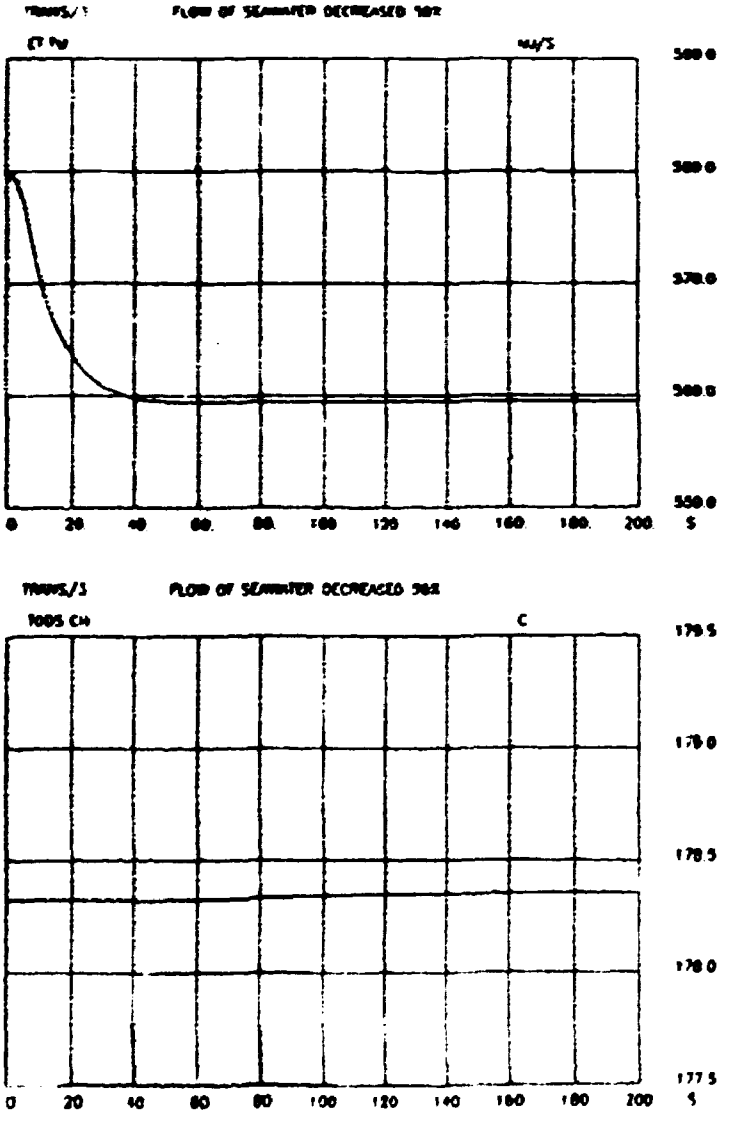


Fig. 4.14. Plot of the third transient: failure of a pump.

5. SIMULATION OF A PART OF A SUGAR FACTORY

5.1. Introduction

In order to demonstrate the broad application of the simulation tool an example from the "chemical" industry will also be simulated. This will be the preheating of a sugar liquor followed by a multiple effect evaporator where the sugar concentration is increased, and it also includes the flash drums. The contents of the submodule library determine the type of process plants that can be simulated easily by using preprogrammed submodels.

The notation that was introduced in 3.6 "Notation for input and output variables" for distinguishing input from output variables was used to quickly construct submodels that will fit together in a complete model of the process plant. This procedure will be described.

The first model that was made was very stiff because the pressure dynamics was included, but since the pressure does not interact directly with the dynamics of the rest of the model the dynamics was replaced by quasi-stationary calculations. How this was done will also be described. It will illustrate how the notation mentioned above can again be used and it will show the ease with which submodules can be replaced with each other.

5.2. Description

Figure 5.1 shows an example of a multiple effect evaporator with preheaters and flash drums as well as boiler, turbine and condenser. Only the steam and condensate flows are shown. The sugar liquor passes the preheaters from right to left before flowing through the evaporators from left to right.

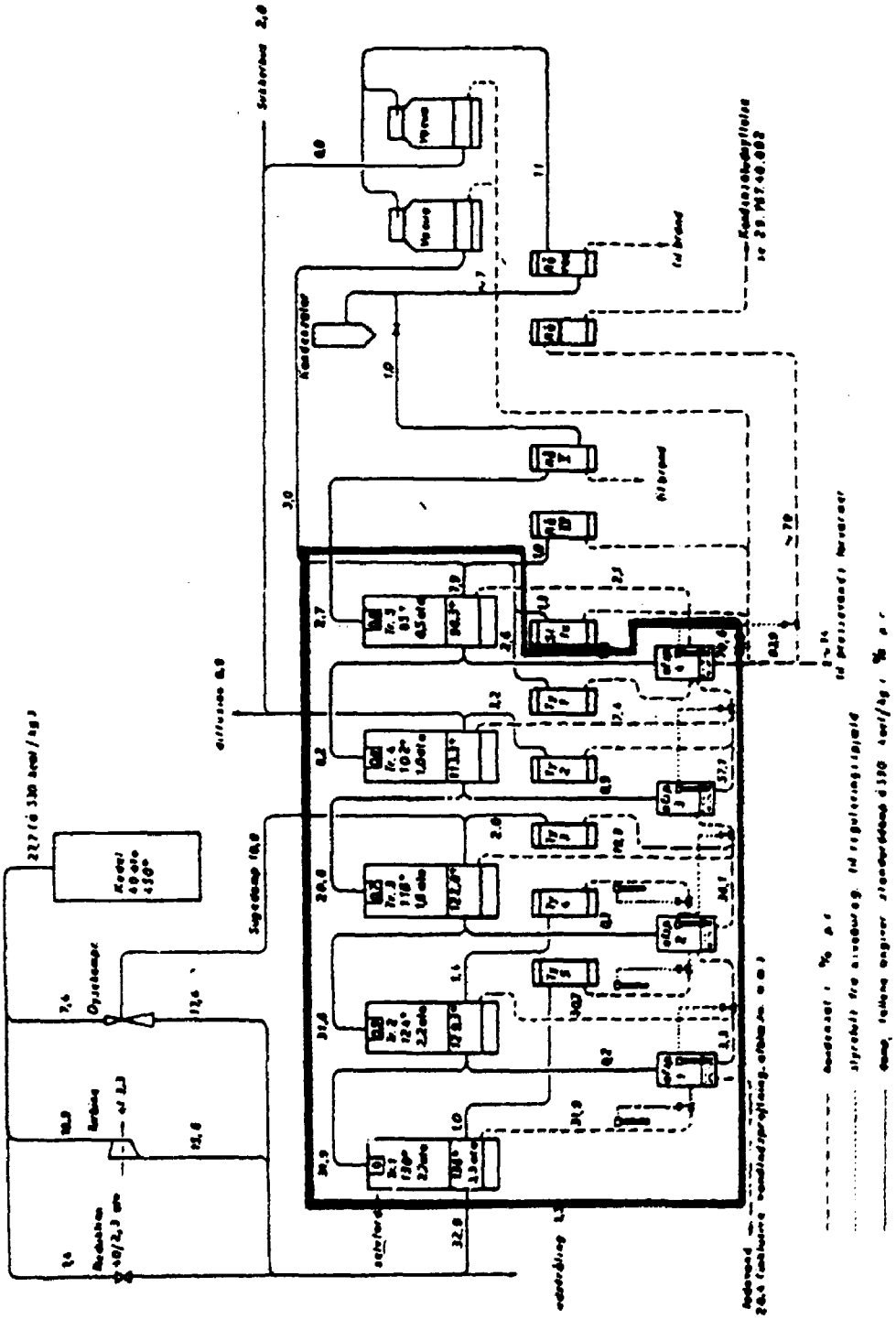


Fig. 5.1. An example of a multiple effect evaporator.

The model to be described here includes the components within the frame: five preheaters, five evaporators and four flash drums. The steam flows (the solid lines) out of the frame are not included in the model. The numbers at the top of the evaporators indicate a flow of steam or vapour being drawn from them and this has been included in the model.

The part of the process plant that is being simulated is shown schematically in Fig. 5.2 which also shows the sugar liquor flows. None of the controllers in Fig. 5.1 is modelled but there are controllers for control of the outlet flow of sugar liquor from the evaporators, for the flow of vapour being extracted at the evaporators and for the inlet flow of sugar liquor to the first preheater.

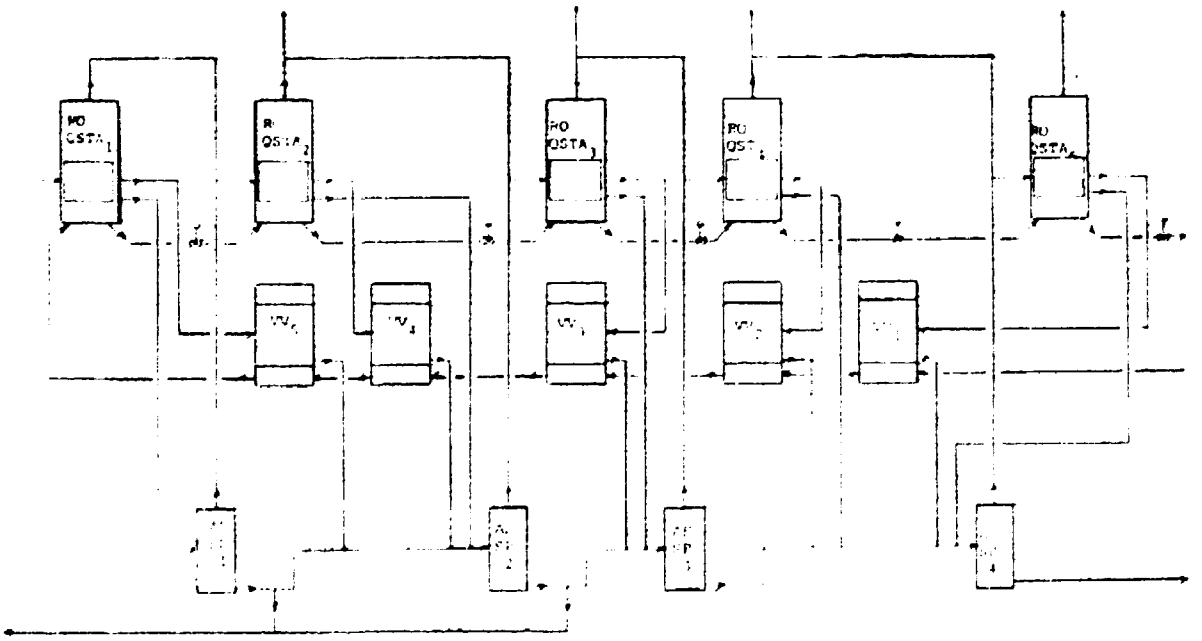


Fig. 5.2. Schematic description of the multiple-effect evaporator with preheaters and flash drums.

The vapour that is extracted is used, for example, for heating purposes elsewhere in the factory. The flow of vapour is controlled so that a given flow of enthalpy is being extracted. This extraction is made only if the vapour pressure is high enough to actually enable extraction of the necessary amount of vapour.

The steam drawn into the first evaporator comes from a boiler. The feedwater to the boiler is drawn from the first or the first two flash drums. The control of this is assumed to be ideal.

The programme is split into two parts. One part contains the components shown in Fig. 5.2 and the other contains the controllers. This division simulates a situation where a control panel contains the controllers and controls the flows through communication with the process plant. The inlet flow of sugar liquor and the extraction of vapour are controlled directly while the flows of sugar liquor out of the evaporators are controlled by valves.

5.3. Multiple effect evaporator module

The multiple effect evaporator extracts steam from the boiler. The boiler is made ideal which means that it is possible to produce steam at exactly the rate needed and at a given pressure. The feedwater flow from the first two flash drums is controlled ideally so that the feedwater flow rate is equal to the extraction flow rate of steam. These flows are attributes of the inlet and outlet flows. They are not used anywhere outside this module but even so they are indicated in Fig. 5.3 which shows all the input and output connections to the module with the notation introduced in 3.6 "Notation for input and output variables".

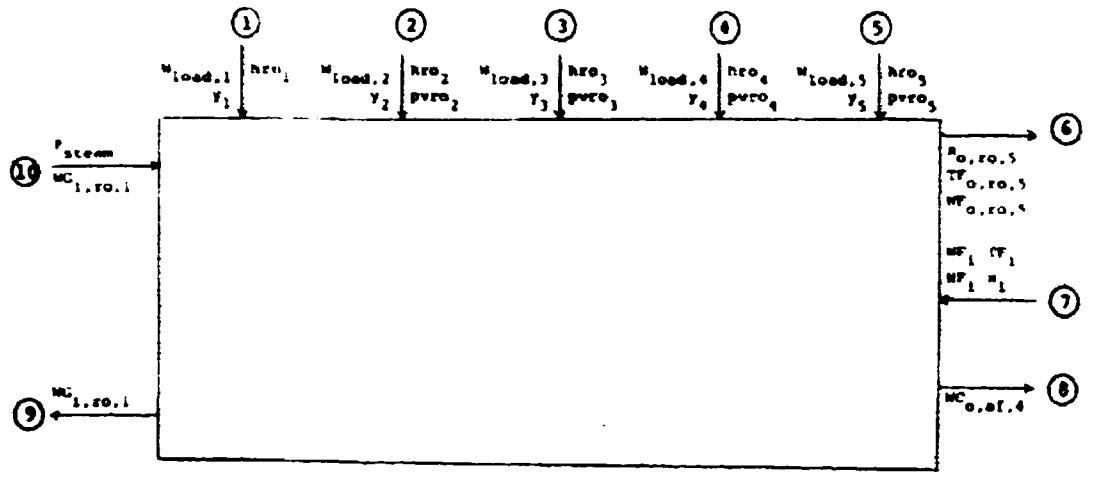


Fig. 5.3. The input and output connections to the multiple-effect evaporator.

The sugar liquor flows in with all the attributes known and it flows out with values for the attributes calculated in the module.

Condensate flows out at a rate calculated in the module while the flow rates of extracted vapour are calculated in the controller module where the valve settings are also calculated. The controller module uses sugar liquor levels and vapour pressures in the evaporators to calculate the valve settings and the extraction flows.

In the first model the pressure dynamics were included, but this introduces a stiff system of differential equations which makes the simulation of a start-up procedure time consuming. Therefore it was decided to use quasi-stationary expressions to calculate the pressure. This is done in other works as well, e.g. (BOLMSTEDT, 1977) and (MÄKELÄ, 1981). In fact no references were found to do it differently.

Most of the model can be made by using the submodule library where there are submodels for evaporators, for preheaters and for flash drums. A few FORTRAN statements are being used to

model valves, calculate pressure and to calculate inlet flow rates and inlet flow of enthalpy to the flash drums as required by the submodule.

5.3.1. Model development

The steam flows through tubes and the resistance in the tubes results in pressure reductions which are calculated from the flow rate. Schematically this is described as shown in Fig. 5.4. According to Bernoulli's equation (HANSEN and SØLTOFT, 1980, p 58) the following expression gives a relation between the pressure difference (P_i-P_o), the steam flow W and the density ρ

$$P_i - P_o = k \cdot W^2/\rho \qquad (5.1)$$

The constant k depends on the tube and it is assumed to be the same for all the tubes since no data were available. k was chosen so as to make the pressure drop of the steam that is flowing from the evaporator in stage 1 to that in stage 2 in a given steady state approximately equal to 0.02 bar.

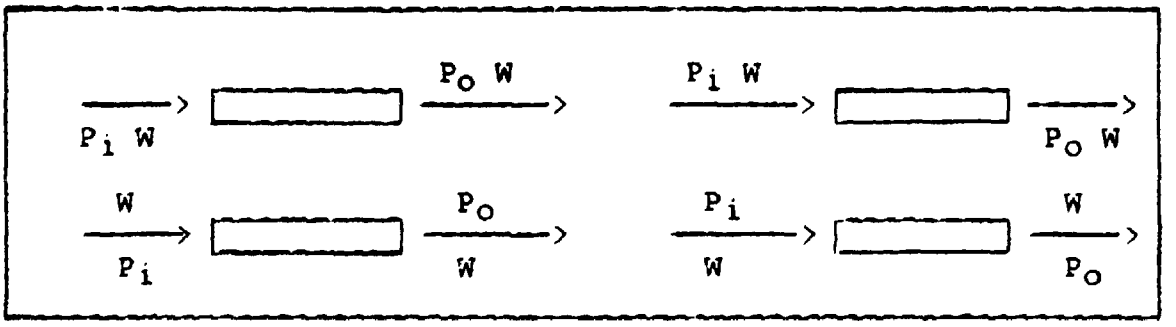


Fig. 5.4. Schematic description of the four cases of pressure calculation from steam flow.

For the valves linear first-order models are used. The valve opening v is described by

$$\dot{v} = (y-v)/\tau \qquad (5.2)$$

where

y is the valve setting and
τ is a time constant.

The flow through the valve is calculated as

$$W = VC \cdot v \qquad (5.3)$$

where VC is the valve constant. The model of a valve is shown schematically in Fig. 5.5.

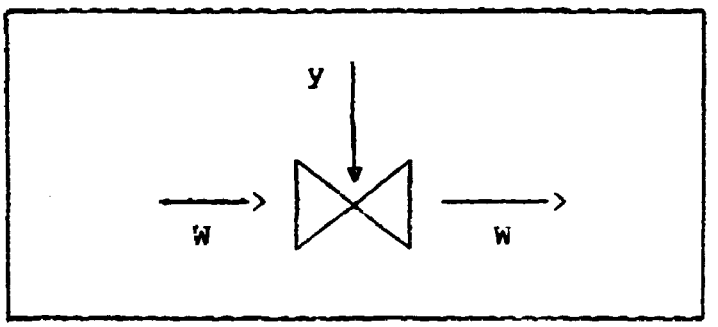


Fig.5.5. Schematic description of a valve model.

5.3.2. Robert evaporator

The evaporators are all Robert evaporators. A flow of sugar liquor enters at the bottom from where it can reach the inside of some tubes. On the outside of the tubes steam is condensing and the heat that is subsequently released causes the evaporation of some of the water from the sugar liquor. This boiling also results in extraction of sugar liquor to a volume above the tubes. While falling down again some of the liquor enters a center tube from where it leaves the evaporator. One vapour flow is extracted for heating purposes in the next evaporator and another flow is extracted for example for heating purposes elsewhere in the factory.

The level of sugar liquor in the evaporator is controlled by the outlet flow of liquor. Some of the steam on the outside of the tubes is used for heating purposes in the preheaters.

The sugar liquor inlet flow rate, specific enthalpy and solute mass fraction are input variables and so is the outlet flow rate. The outlet enthalpy and solute mass fraction can be calculated from conservation of energy and of mass considerations.

The rates of the two flows of vapour leaving the evaporator are input variables. By assuming that the vapour and the liquor is at saturation the pressure and specific enthalpy of the vapour can be calculated. The elevation of the boiling point is taken into account.

The pressure and specific enthalpy of the steam entering the evaporator are input variables and so is the flow rate of steam extracted by the preheater. Applying conservation of energy will give the flow of condensate which is used for calculation of the inlet flow rate of steam using conservation of mass.

It is assumed that the steam pressure of the extracted steam is the same as the pressure of the inlet steam. Some of the steam is condensing so it is at saturation and the specific enthalpy of the steam and of the condensate can be found.

Even though the process of boiling means extraction of sugar liquor to the volume above the tubes it will be assumed that it is possible to use a well-defined sugar liquor level.

This model of the evaporator is shown schematically in Fig. 5.6. The details of the model are described in Appendix F; the name of the submodule is ROQSTA.

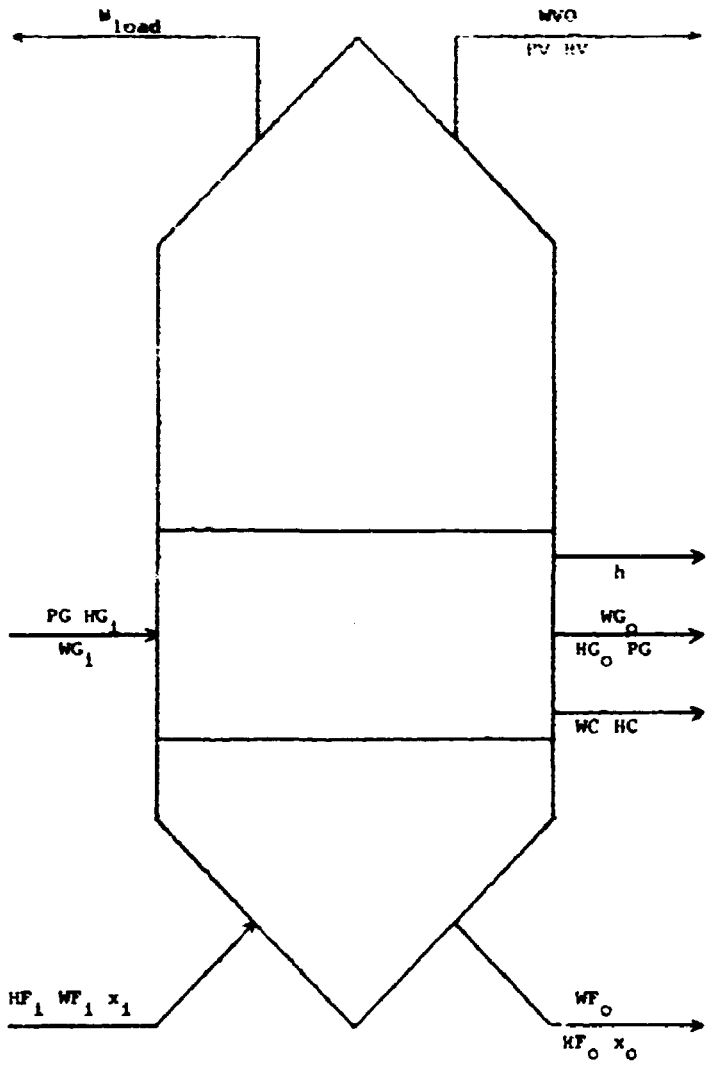


Fig. 5.6. Schematic description of a Robert evaporator model.

5.3.3. Preheater

In the preheater a flow of sugar liquor is heated by condensing steam. The sugar liquor flows inside some tubes and the steam condenses on the outside. The condensate flows out of the preheater.

The attributes of the inlet flow of sugar liquor are input variables and the attributes of the outlet flow are output variables. The outlet flow rate is the same as the inlet flow rate. The specific enthalpy and the solute mass fraction are calcu-

lated by using conservation of energy and of mass. The temperature is calculated from the enthalpy and solute mass fraction using a sugar function, i.e. a thermodynamic function for sugar liquor.

The pressure and the specific enthalpy of the steam are input variables. Since the steam is at saturation when it is condensing the specific enthalpy of the condensate can be found. By applying conservation of energy the inlet flow rate of steam can be calculated and this is equal to the flow rate of condensate.

This model of the preheater is shown schematically in Fig. 5.7. The details of the model are described in Appendix F, the name of the preheater is VV.

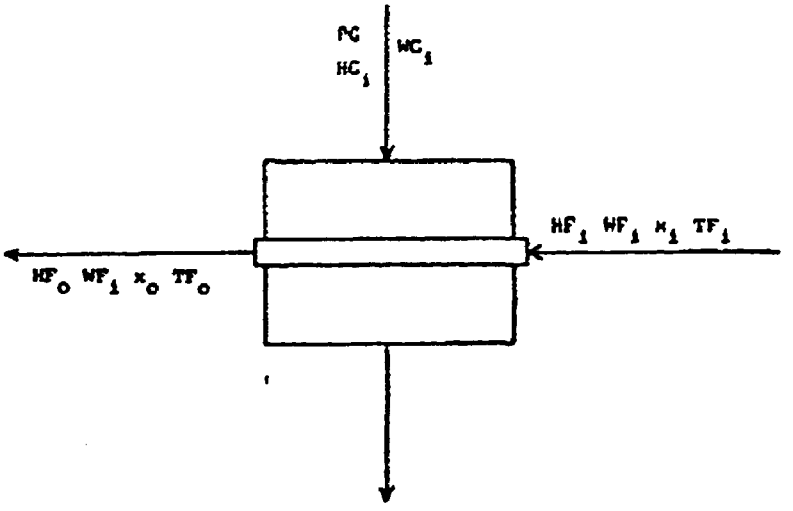


Fig. 5.7. Schematic description of a preheater model.

5.3.4. Flash drum

The flows of condensate enter some flash drums where the pressure is below the saturation pressure which leads to a flash evaporation. There is a control of the level of condensate in the flash drum by an outlet flow of condensate.

The total flow rate and the total flow of enthalpy to the flash drum are input variables. Applying conservation of energy the specific enthalpy of the condensate in the flash drum can be

calculated. The steam pressure is an input variable and the degree of superheating leads to an evaporation rate calculated by using an empirical expression. By applying conservation of mass and approximating the level control by an ideal controller the outlet flow rate of condensate can be calculated. This model of the flash drum is shown schematically in Fig. 5.8. The details of the model are described in Appendix F; the name of the submodule is AFSP.

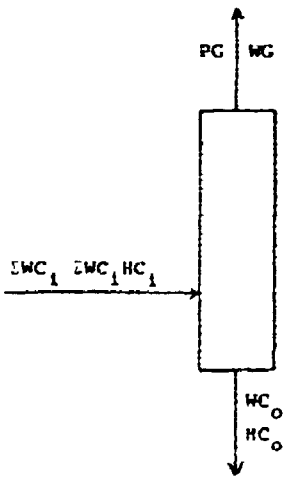


Fig. 5.8. Schematic description of a flash drum model.

5.3.5. Connecting the submodules

In this subsection the notation that was introduced in 3.6 "Notation for input and output variables" will be applied to demonstrate that the models as they have been described indeed do fit together to form a complete model of the part of the sugar factory that is going to be simulated. The three models of the submodules are shown schematically in the previous subsections and so are the models of steam flow through a tube and sugar liquor flow through a valve (see Figures 4.4-8).

Several flows of condensate are mixed in a flash drum but the submodule has only one inlet flow. Therefore a schematic representation of a model for mixing condensate must be available. This is shown in Fig. 5.9.

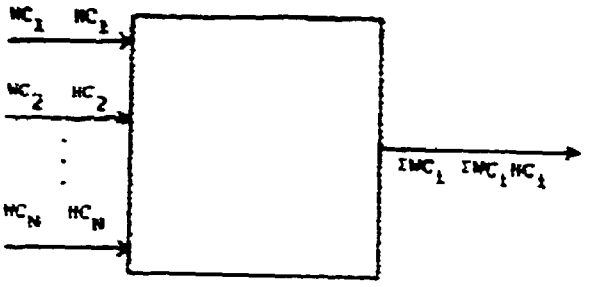


Fig. 5.9. Schematic description of mixing flows of condensate.

From the first two flash drums condensate can be extracted as feedwater to the boiler. The rest of the condensate flows on to the next flash drum so a condensate splitter is needed and this model is shown schematically in Fig. 5.10. The flow rate of feedwater WF_o that is needed is part of the input to the splitter and so is the flow rate of condensate that is available. The flow rate of feedwater WF is either WF_o or WC_i whichever is the smaller. The condensate flow-rate out WC_o is the difference between the inlet flow rate WC_i and the flow rate WF of feedwater.

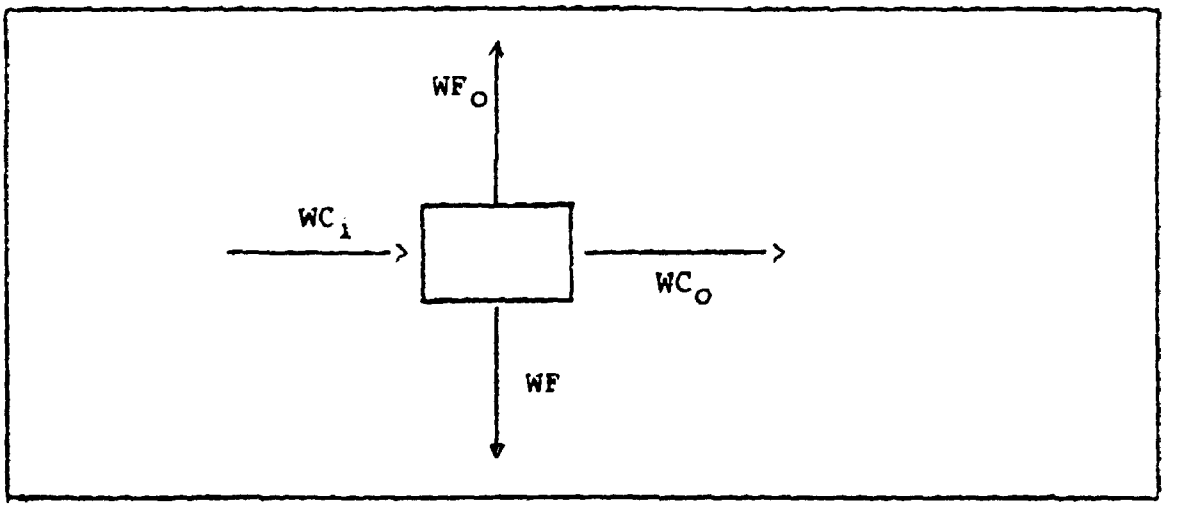


Fig. 5.10. Schematic description of a condensate splitter model.

Finally a schematical description of the mixing of steam must be made. Steam from a flash drum together with vapour from a Robert evaporator makes up the flow of steam being extracted by the next Robert evaporator. A known flow-rate of steam comes from the flash drum and the rest of the steam is extracted as vapour from the previous evaporator. The schematic description of this model for a steam mixer is shown in Fig. 5.11.

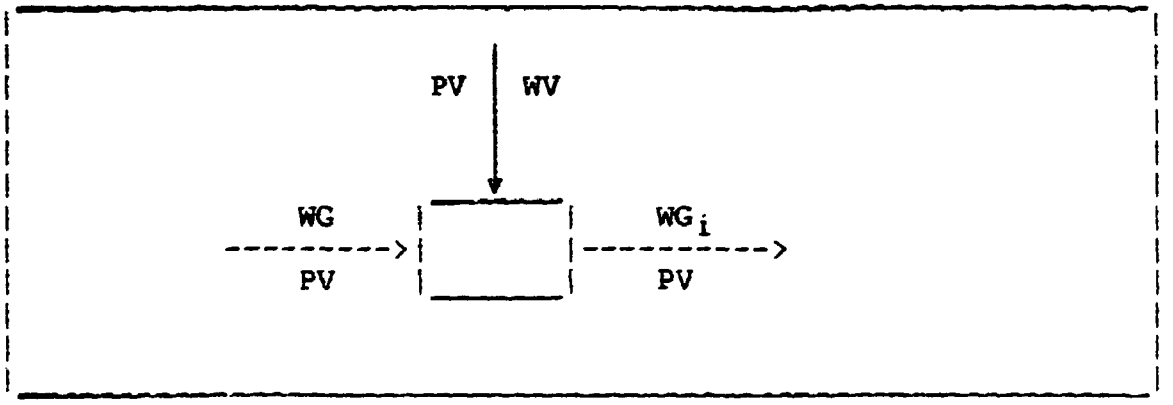


Fig. 5.11. Schematic description of a steam mixer model.

By reading this and the previous subsections it may be possible to see that the models do in fact fit together but it is hard to do so convincingly. The schematic description of the models can be drawn in one figure and where a flow from one model is connected to a flow of another the user can see that (or whether!) the models fit together by noticing that corresponding attributes of the flows are written on opposite sides of the arrows. If an arrow is bent input variables must be moved from above an arrow to the left-hand side of the arrow and vice versa.

This is done in Fig. 5.12 for a part of the multiple effect evaporator; this part is big enough to show that all the models fit together. Numbers in circles are written for input variables that are input to the module and for output variables that are not used here but in the other module. These numbers are the same as those in Fig. 5.3.

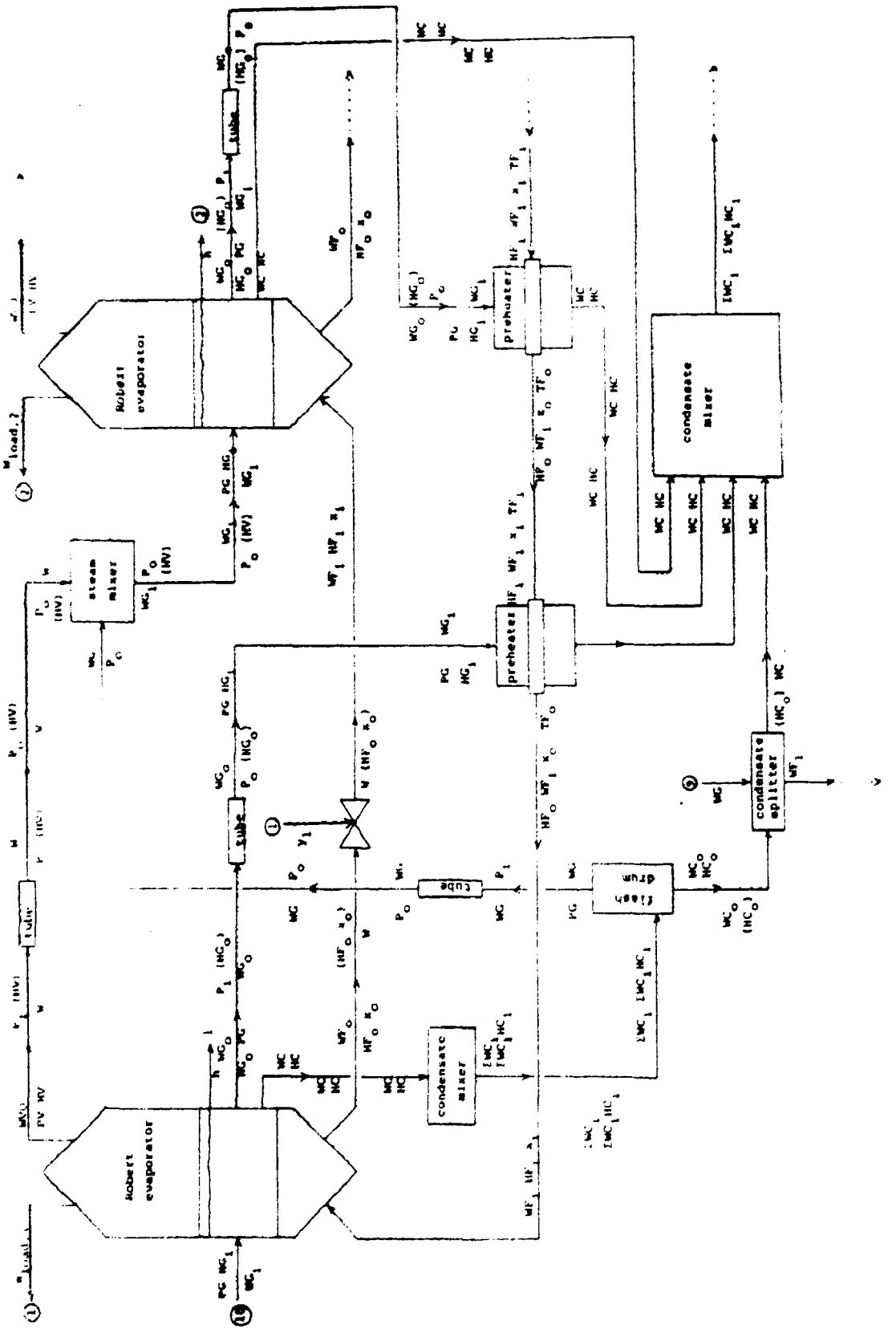


Fig. 5.12. Connections of schematic descriptions of submodels for a part of the multiple effect evaporator.

It is rather time consuming to draw drafts like Fig. 5.12 but so is the process of gaining an overall view of the model being developed. Figure 5.12 was a help to demonstrate clearly that the submodels together constitute a consistent system of equations for the complete model.

In Fig. 5.12 some of the attributes of the flows are written in parentheses. These attributes are not actually part of that particular model and thus they are just being passed on unchanged from being input variables to being output variables. An example of this is the specific enthalpy of steam flowing through a tube. The enthalpy is not part of the model of the tube but it is needed as an input variable to the model of the Robert evaporator and the preheater.

Even though this methodology was developed while setting up this model of the multiple effect evaporator it took a surprisingly short time to make a consistent model. Looking at Fig. 5.12 it may look complicated but indeed it does represent a lot of information and it would be rather more surprising if it looked simple. The methodology has to prove itself by further tests, however.

This way of presenting connections between submodels almost immediately gives rise to the idea of making the connections by using graphical tools while programming a model. This has been done in other simulation systems (see for examples (BROWNE, DUTTON and NEUSE,1986) and (SCHAAK, BARTELLS and LONG, 1986)) but the complexity of Fig. 5.12 is a problem that must be dealt with in an efficient manner if it is going to be possible to construct a graphical presentation of a module like this one on a graphical screen.

5.4. Controller module

The controller module contains the models of all the controls of the multiple effect evaporator module. There are three types of control:

1) sugar liquor level in the Robert evaporators
2) vapour extraction from the Robert evaporators
3) inlet flow rate of sugar liquor.

A schematic description of the controller module is shown in Fig. 5.13. It is seen that the five connections between this and the multiple effect evaporator module fit together. The other connections in both of the modules are established in the connecting system.

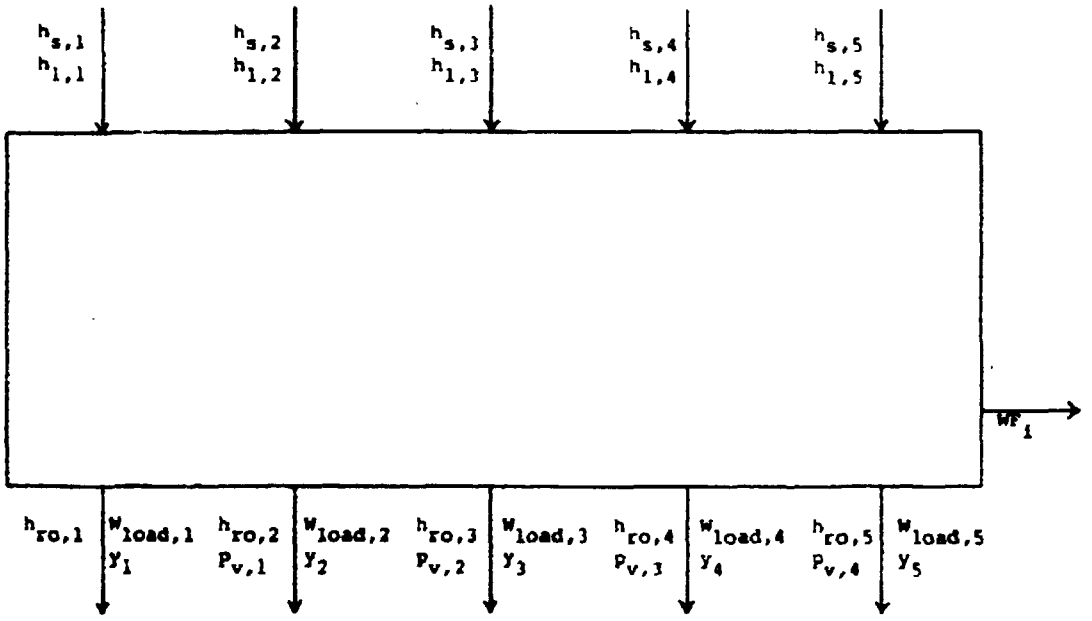


Fig. 5.13. The input and output connections to the controller module.

5.4.1. Level control

The sugar liquor level is controlled by the outlet flow rate which in turn is determined from the valve opening. It is the valve opening that is being controlled by the controller. The controllers were chosen to be proportional controllers.

There were no data available for the P-controllers but they were proposed to be made "good" (JØRGENSEN, 1986). This will be described in 5.5.2 "Parameters for the controllers".

5.4.2. Vapour flow control

As is shown in Fig. 5.2 vapour is being drawn from the last four evaporators. The vapour is used for heating purposes elsewhere in the factory and a given enthalpy flow-rate is therefore needed.

In working conditions this procedure is straightforward, but in situations such as start-up care must be taken. The driving force establishing the vapour flow is the pressure gradient which therefore must be built up before any vapour can be extracted, and therefore a switch must be used for the closure of the valve if the pressure is too low.

To this end a submodule was designed to switch between two states using a ramp. The name of the submodule is RAMPSW. If the pressure drops below a lower value, the submodule sets up some variables that can be used for making a valve close controlled by a linear ramp. If at a later time the vapour pressure increases beyond a higher value the variables are changed to make the valve open again.

The submodule RAMPSW uses some facilities in the runtime executive DYSIM86 that makes it possible to detect the state event with a given accuracy, i.e. the time steps are iteratively decreased until the event, here the vapour pressure crossing a value, is determined with adequate accuracy.

The variables that are calculated by the submodule are the initial and final values for the ramp, PI and PT, and the time span, TI to TT. From these variables the set point P_S can be calculated with this ramp function

$$P_S = PI+AMIN1(1.,((T-TI)/(TT-TI)))*(PT-PI)$$

where

T is the present time
AMIN1 is a FORTRAN function returning the value of the variable that is the least.

The controllers were chosen to be PI-controllers since they result in no offset. The actual data for the controllers were found by making them "good" (see 5.5.2 "Parameters for the controllers").

5.4.3. Sugar liquor inlet flow control

It is necessary to control the inlet flow of sugar liquor because of the following situation: Imagine that the enthalpy flow rate from Robert evaporator number 2 is increased. In order to maintain the sugar liquor level in the evaporator the outlet flow rate is reduced. This will also be done for the rest of the evaporators by the level controllers there. The inlet flow rate to the last evaporator may in some cases become too small and the valve will close completely, but the level will continue to fall because of the evaporation. This situation is unacceptable and it can be avoided by controlling the inlet flow rate of sugar liquor to the multiple effect evaporator.

Since it is the liquor levels that are important they are used for the control of the inlet flow. A PI-controller is chosen to do so. A measure of the accumulation is the difference between the measured levels and their set points. These differences are summed with weight factors. If the weight factors are chosen to be the area of liquor in the evaporators the weighted sum is directly the volume being accumulated. This was chosen in this simulation.

Now the levels are used for two different controllers which is a potentially unstable situation. In the simulations executed so far no problems were seen.

5.5. Parameters

5.5.1. Parameters for the multiple effect evaporator

There are nine evaporators in the actual installation. Eight of these work in pairs while there is only one in the fifth stage of the multiple effect evaporator (DDS,1985). The steam and vapour pressures in a pair of evaporators is the same in both but the sugar liquor enters one evaporator before the other and there will be different concentrations, temperatures and levels. The two evaporators in a pair may be very different but even so each pair was lumped to one component.

The data that are needed have been calculated from JØRGENSEN, 1986. The necessary data are the parameters to the submodules, valve constants, etc. In Tables 5.1 and 5.2 the parameters for the evaporators and the preheaters are compiled.

AUMAX	MW/K	15.73	12.86	7.572	2.291	0.5664
HOPT	m	1.5	1.5	1.5	1.5	1.5
CT	MJ/K	28.5	3^.2	25.2	11.0	7.99
A	m2	10.21	10.27	7.526	5.237	2.648
VU	m3	17.2	17.2	10.0	6.4	3.2

Table 5.1.

K	MW/K	1.125	1.365	1.899	1.962	1.243
CT	MJ/K	3.55	6.57	6.57	6.57	2.09
FVOL	m3	2.119	3.89	3.89	3.89	1.238

Table 5.2.

The parameter HOPT is the optimal sugar liquor level for heat transfer and at optimum the heat transfer coefficient is AUMAX.

The nonlinear function relating the actual heat transfer coefficient to the level is assumed to be the same for all the evaporator components. It has been approximated from data for a particular study (JØRGENSEN, 1986) and only the dependence on the level has been considered. CT is the heat capacity in the tubes, A the surface area of the liquor, and VU the liquor volume under the tubes.

KGT and KTF are the heat transfer coefficients from the steam to the tubes and from the tubes to the liquor in the preheaters. CT is the heat capacity in the tubes and FVOL is the volume of liquor in the tubes in the preheaters.

For the flash drums only the volume V of condensate is needed and it is the same for all the flash drums : V = 1.8 m^3.

The valve constants were chosen so that the valve openings are reasonable at working conditions and the values are listed in Table 5.3. The valves are modeled as a first-order system with time constants which are also listed in Table 5.3.

VC	kg/s	153.3	153.3	102.2	100	75
τ	s	1	1	1	1	1

Table 5.3.

For calculating the pressure drop the relation (5.1) is used. For conditions considered to be "normal" the coefficient k was determined. It is assumed that k can be used for all tubes causing pressure drops in this simulation. The value for k is $1.78 \cdot 10^{-5}$ and it is called PRHOW in the module.

5.5.2. Parameters for the controllers

First the controllers for the sugar liquor levels are considered. The controllers were dimensioned by linearization and only looking at the influence on the level from the outlet flow. In this way a second order system is found. The proportional band of the controller was chosen by demanding 45° phase margin. The parameters for the Robert evaporators and the valves, Tables 5.1 and 5.3, were used for calculating the proportional bands.

The input ranges were chosen to be [1.25 m,1.75 m] because of the rather good control. The output range is [0,1] and the offset for the output were chosen to be close to the working conditions because this will minimize offset due to the usage of P-controllers. The output offset Y_s and the proportional bands are compiled in Table 5.4.

Y_s	kg/s	0.72	0.446	.04249	0.3506	0.3951
P_b		0.030	0.03	0.027	0.038	0.057

Table 5.4.

The vapour flow controllers were also dimensioned by linearization and by assuming negligible influence from everything except flow changes. Now the proportional band P_b was calculated by choosing the amplitude ratio to be 0 dB when the phase is -45°. The reset time for the PI-controllers were all chosen to be 2 s. The rest of the parameters are compiled in Table 5.5. The input range is [XMIN,XMAX] and the output range is [YMIN, YMAX].

XMIN	MW	-----	0.0	0.0	0.0	0.0
XMAX	MW	-----	55.0	55.0	15.0	30.0
YMIN	kg/s	-----	0.0	0.0	0.0	0.0
YMAX	kg/s	-----	30.0	30.0	15.0	10.0
PB		-----	0.0493	0.0491	0.1789	0.0884

Table 5.5.

The sugar liquor inlet flow controller was chosen completely arbitrarily. The weight factors mentioned in subsection 5.4.3 were chosen to be the sugar liquor surface area so the weighted sum is the volume of accumulated liquor. The input range is chosen to be [-50 m^3, 50 m^3] and the output range is [0 kg/s, 200 kg/s]. The reset time for the controller is 1 s and the proportional band is 0.03.

5.6. Simulation

Three simulation runs are going to be described here: First a set point change in an evaporator, then a shut-down procedure and lastly a start-up procedure. Two steady states were needed, one for the working condition and shut-down procedure and one for the start-up procedure.

There were no data available for all the necessary state variables in the multiple effect evaporator from which the data for the components were compiled but some steady state values of another multiple effect evaporator were available (JØRGENSEN, 1985) and the limitation of the number of components and of the external connections were also inspired from that work. So even though the data for the components differ somewhat and the connections between the flash drums and the other components in the multiple effect evaporator may not be the same this steady state was sought. This can of course be done only by choosing

the values of the input variables in our model as they are in the steady state being sought. These input variables are the inlet flow rate, temperature and solute mass fraction of the sugar liquor, the temperature of the steam being extracted from the turbine and the external extraction of steam at the last four stages. The external extraction of steam cannot be given exactly since the input variable is actually the enthalpy flow which is dependent on the vapour pressure, and since this is a derived attribute there will be a difference between the two sets of variables; but the best one can do is to use the enthalpy flow calculated from the steady state being sought as the input variables in our simulation.

Even with all these discrepancies the steady state found in this simulation is astonishingly close to the steady state in (JØRGENSEN, 1985) as it is seen in Table 5.6 where a few of the variables are compared.

The first simulation to be discussed is a set point change in the first evaporator. This illustrates that the controllers have indeed been chosen to be good (see Fig. 5.14). The sugar liquor level in the evaporators, HRO, shows little overshoot and no oscillations. It is seen that since the first evaporator is being filled in order to adjust to the new set point there is less sugar liquor flowing into the next evaporators and therefore the levels there are falling until the level in the first evaporator has reached the new set point. The inlet flow rate, WFI, of sugar liquor is also seen to be controlled rather well. The perturbations on the temperatures are rather small.

The next simulation is a shut-down procedure (Fig. 5.15). This is done by reducing the solute mass fraction to (almost) zero and reducing the temperature of both the sugar liquor and the steam to 30°C. The reason for choosing 30°C is that the thermodynamic functions that are available have 20°C as the lowest applicable value. By reducing the steam temperature to 30°C the evaporators will extract no steam since this is below (and eventually equal to) the tube temperature, and therefore no heat transfer from the steam to the tubes is possible. The sugar

Sugar liquor, outlet, preheaters;
Temperature (C)

1)	98.0	110.5	122.6	131.6	137.1
2)	97.8	110.4	122.5	131.4	137.0

Sugar liquor, outlet, evaporators;
Temperature (C)

1)	134.6	126.8	118.7	109.1	90.2
2)	134.4	126.7	118.7	109.0	89.7

Solute mass fraction (kg/kg)

1)	0.178	0.289	0.461	0.577	0.688
2)	0.179	0.291	0.464	0.582	0.692

Flow rate (kg/s)

1)	110.3	67.9	42.6	34.0	28.5
2)	111.9	68.7	43.1	34.4	28.9

Table 5.6. Comparison of state variables as found in (JØRGENSEN, 1985), 1), and in this simulation, 2).

liquor level in the first four evaporators are controlled quite well but this is certainly not the case for the fifth. This is because the external extraction of steam from the evaporators is shut off whereby more liquor initially must be flowing out to the next until the control of the sugar liquor inlet flow results in a reduced flow rate. The last valve is not big enough to be able to let the surplus pass. The vapour pressure in the last two evaporators were also recorded (Fig 5.15). It is seen that in the last evaporator the vapour pressure is fluctuating quite a bit. This is because the parameters for the submodule

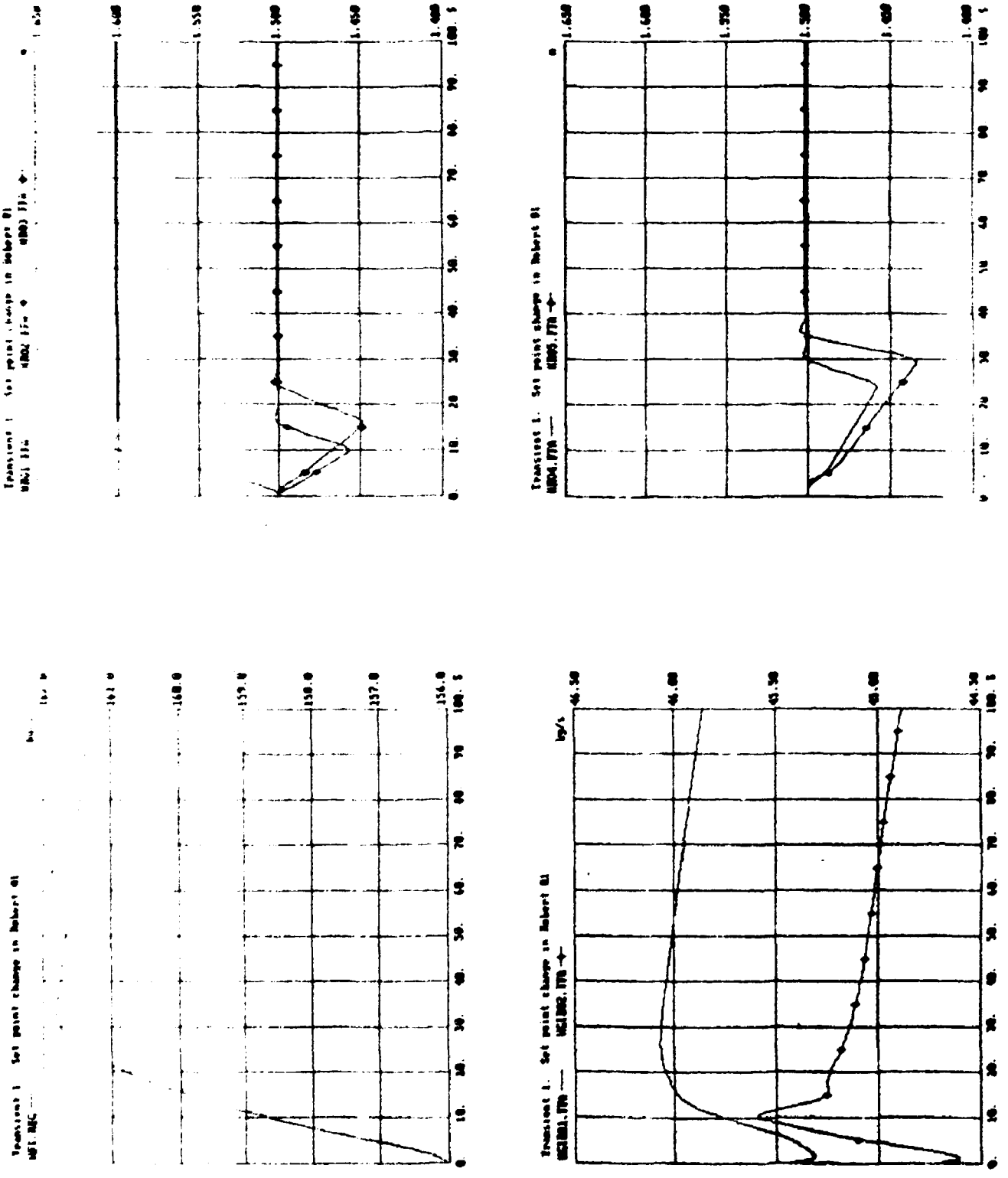


Fig. 5.14. Plot of the first transient: set point change.

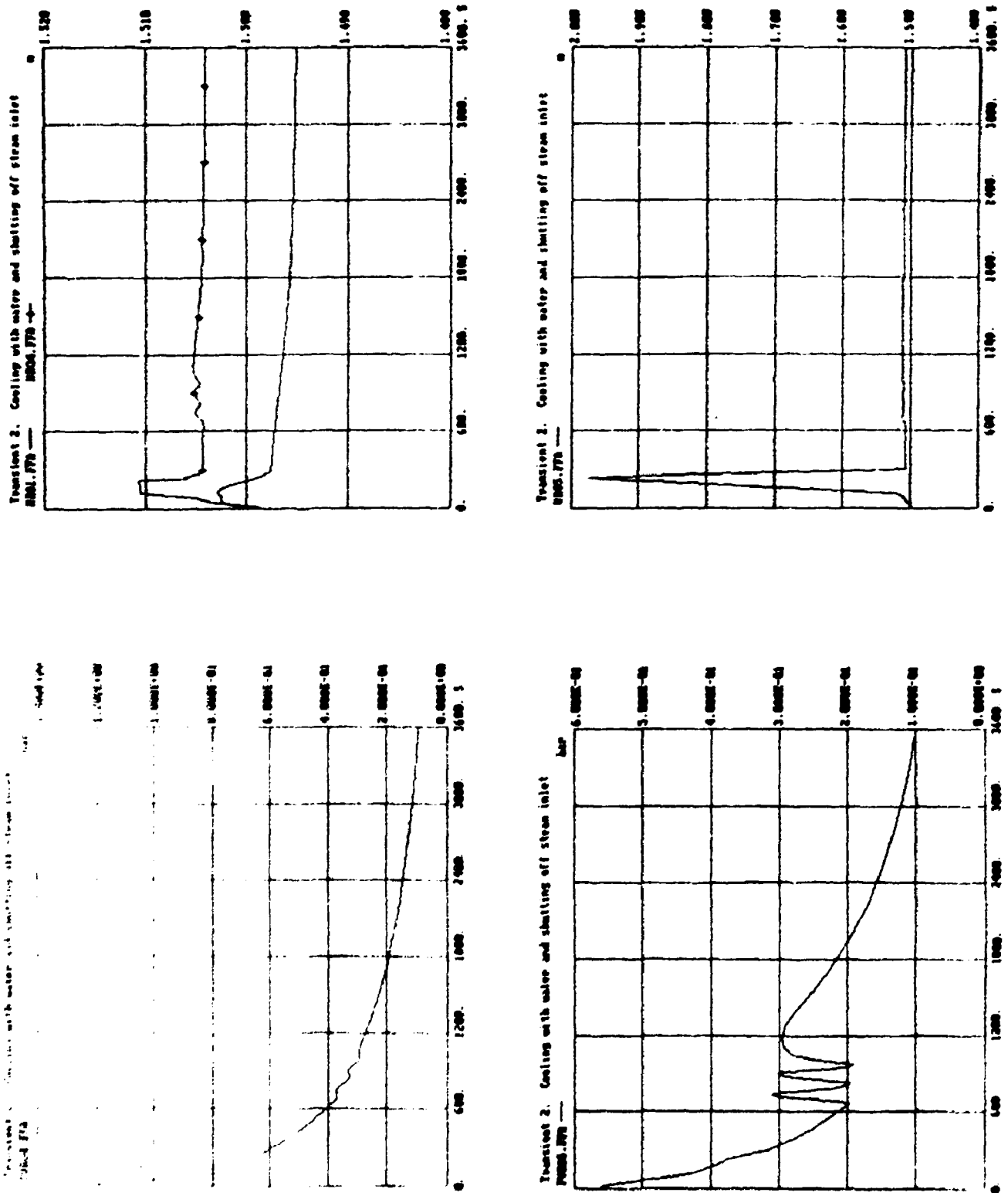


Fig. 5.15. Plot of the second transient: a shut-down procedure.

that controls the external extraction of steam were not chosen satisfactorily. The extraction is turned on and off a few times which leads to the fluctuations in vapour pressure.

In the last situation (Figs. 5.16a-b) a start-up procedure is simulated. First the steady state was found. This is a state where the sugar liquor at inlet is 30°C and the solute mass fraction is (almost) zero. The steam temperature is also 30°C and there is no external extraction of steam. The transient is initiated by increasing the temperature of the steam and of the sugar liquor as well as the solute mass fraction. The levels of sugar liquor are initially kept at their set points but when the conditions are right for extraction of steam from the first evaporator by the second evaporator the levels are disturbed. The vapour pressures and the external extraction of vapour is seen to be smooth.

5.7. Introduction of pressure dynamics

As was described earlier the pressure dynamics were actually introduced from the start and the symbols for the components that were used for convincingly demonstrating that the sub-models fit together, a figure much like Fig. 5.12, was constructed for this purpose, and this figure will be seen to be a little more complicated. The reason for this is that in order to get a reasonable time step in the integrations the pressure dynamics were calculated by lumping together compartments which are at much the same pressure. The pressure drops in the tubes were also used at that time.

The compartments that are lumped together are those with

vapour in a Robert evaporator
steam in the next Robert evaporator
steam in the preheater following this last evaporator
steam in the flash drum, flowing to this last evaporator.

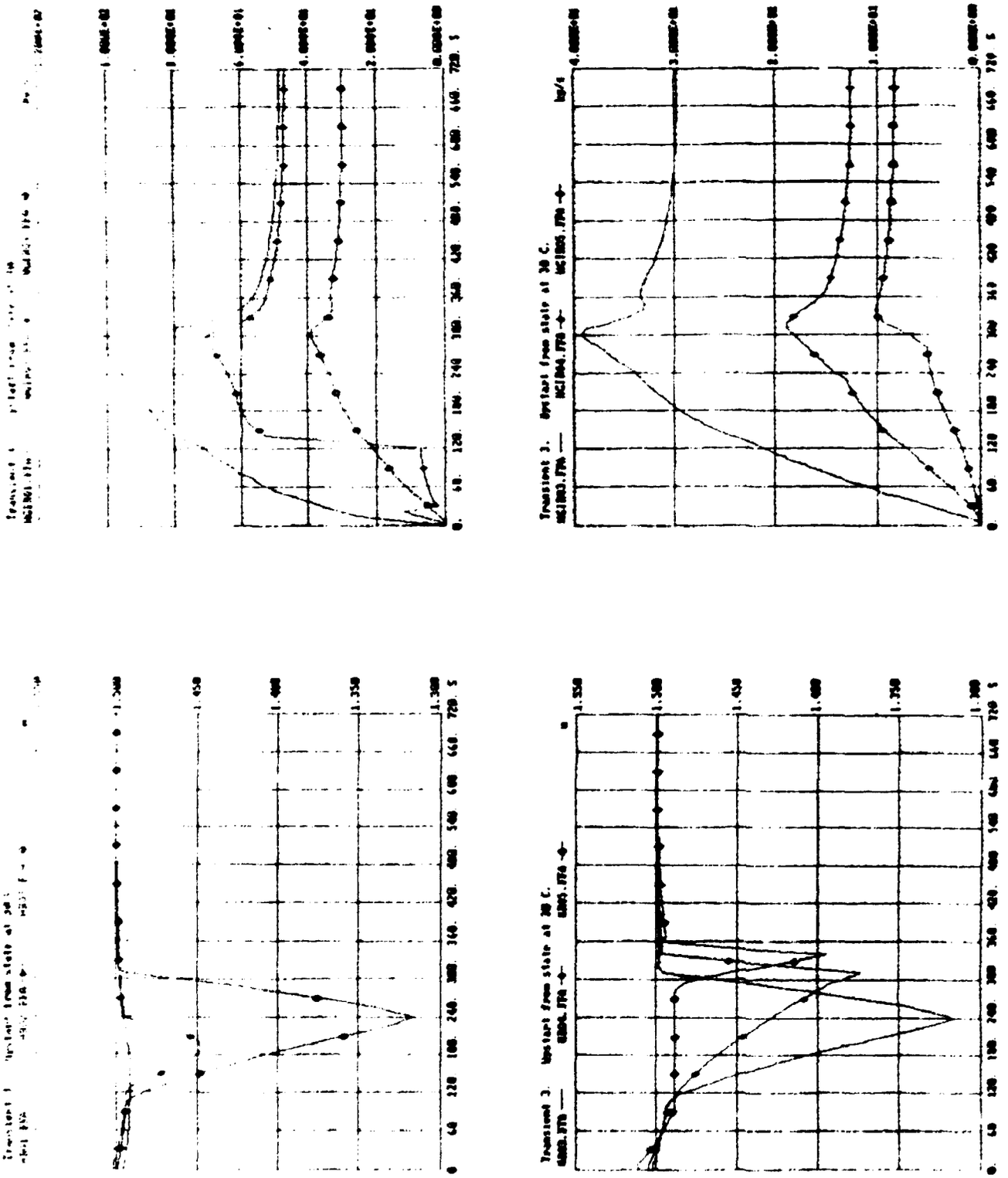


Fig. 5.16a. Plot of the third transient: a start-up procedure.

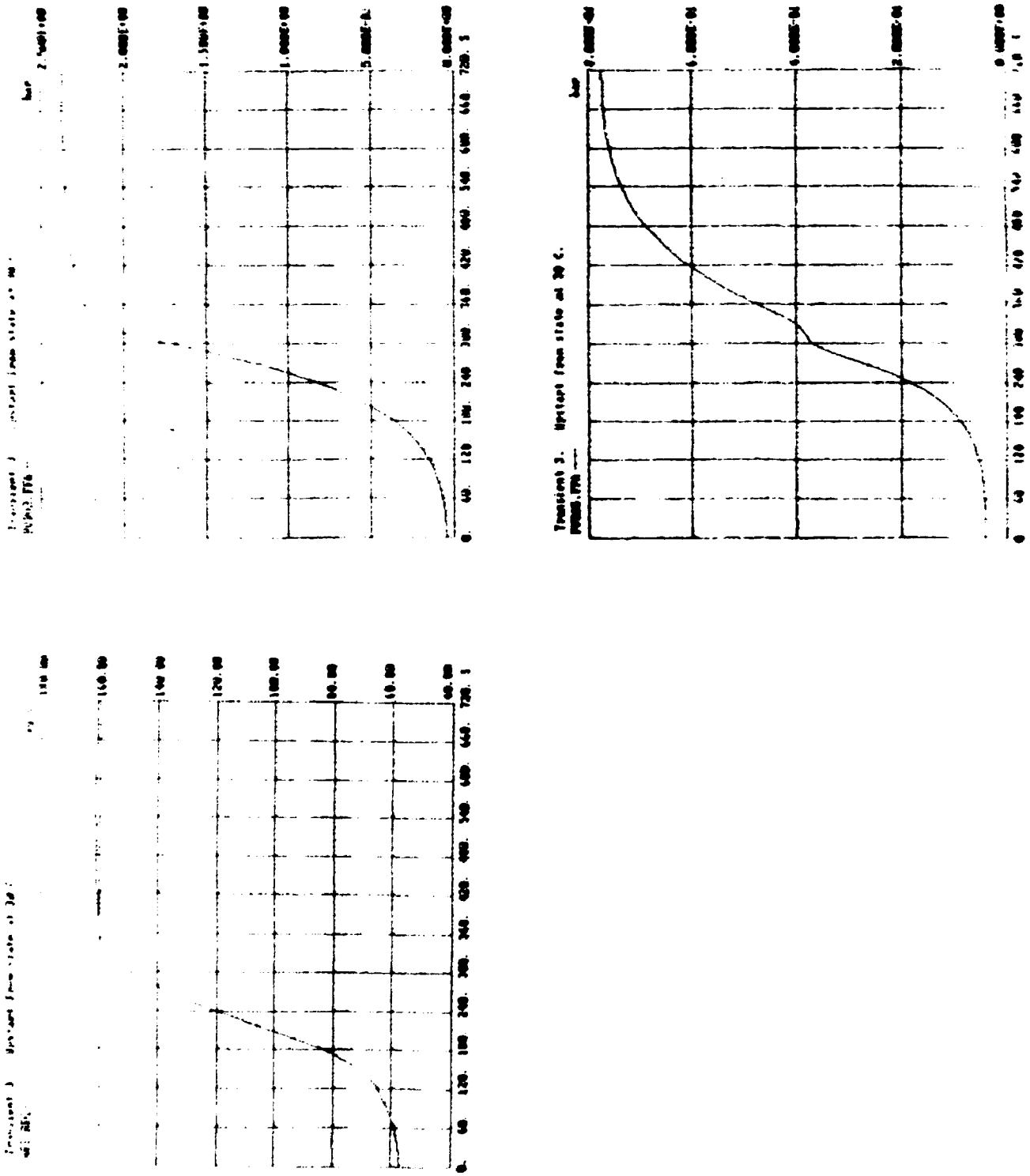


Fig. 5.16b. Plot of the third transient: a start-up procedure.

The pressure that is going to be calculated is related to the steam in the second of the two Robert evaporators mentioned above since it is the "middle" compartment. We will need new schematic presentations for the evaporator and for the tube (see Figs. 5.17 and 5.18). The pressure, PG, that is going to be calculated is an input variable to the submodels.

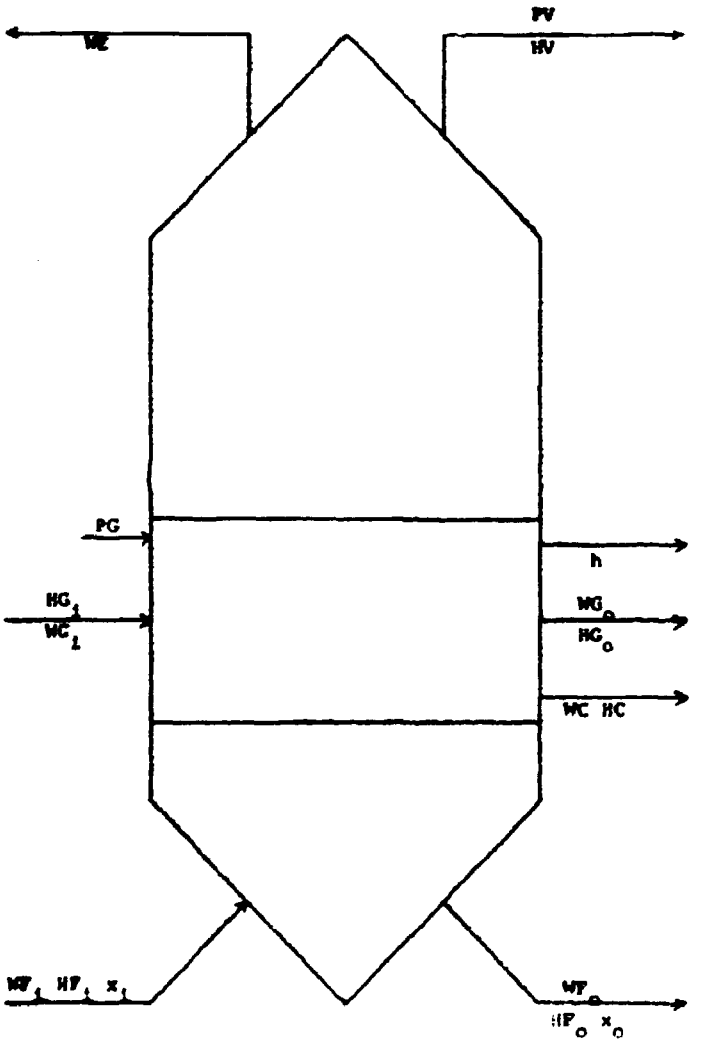


Fig. 5.17. Schematic description of Robert evaporator with dynamic calculation of the pressure.

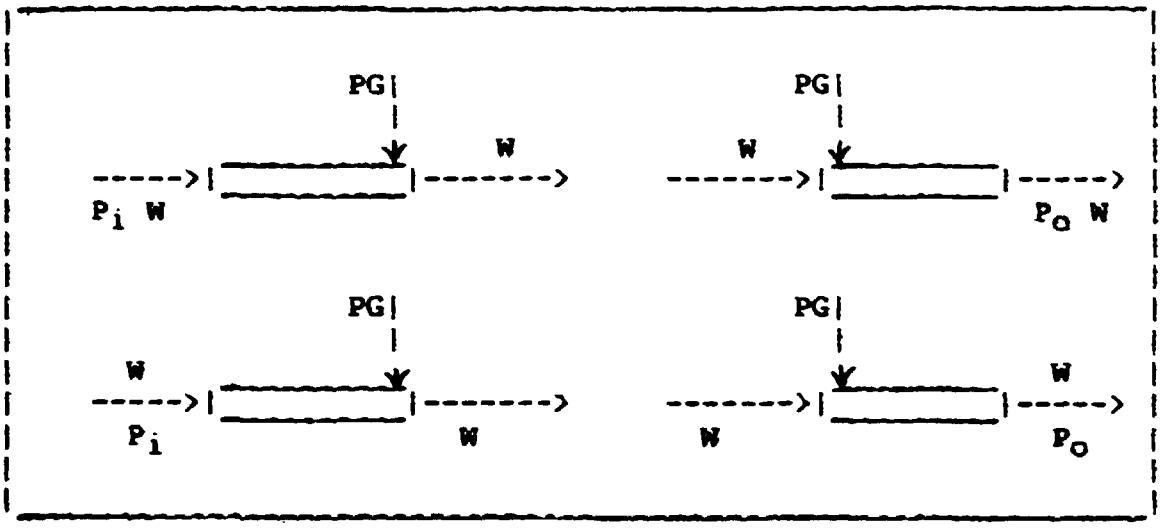


Fig. 5.18. Schematic description of the four cases of pressure calculation from steam flow with dynamic calculation of the pressure.

The pressure dynamics are introduced by applying conservation of mass on this one "compartment". To this end the net inlet flow rate must be calculated. The steam entering the "compartment" comes from

the vapour being evaporated in a Robert evaporator
the steam being evaporated in a flash drum

and the steam leaving the "compartment" is

external extraction of vapour on a Robert evaporator
steam condensing in the next Robert evaporator
steam condensing in the preheater following this evaporator

The net inlet flow rate is calculated from the flows above. These flows are all output variables from the submodels. The pressure dynamics are given by

$$\dot{PG} = \frac{\Sigma W_i - \Sigma W_o}{V \rho_g'} \qquad (5.4)$$

where

ΣW_i is the inlet flow rate of steam to the "compartment"

ΣW_o is the outlet flow rate of steam from the "compartment"

V is the total volume of the "compartment"

$$\rho_g' = \frac{d\rho_g}{dPG}$$

ρ_g is the density of the steam.

The steam flows within the compartments that have been lumped together for pressure calculations which are not used for any purposes such as calculation of pressure drops, need no longer be part of our description. The vapour flow rate out of an evaporator is therefore no longer present and neither is the external extraction of vapour (see Fig. 5.17).

As can be seen only minor changes have been introduced. The pressure is still an input variable to the tube submodel, only now it is calculated elsewhere. In the evaporator the pressure becomes an input variable and the external extraction of vapour is no more interesting. Instead of this the rate of vapour evaporation is calculated from the degree of overheating (see Appendix F). The only changes that have to be made are then: use another submodule with the new evaporator, change a few tube calculations and introduce the pressure dynamics by four new state variables. This way of calculating the pressure cannot easily be indicated in the schematic description (see Fig. 5.19), so this may be regarded as new type of input variables to the model. This problem arises because we are trying to improve the effectiveness of the simulation programme by making the model more complicated. The problem will disappear if the pressure dynamics was calculated on the basis of each of the volumes in the components since the model then resembles the process plant.

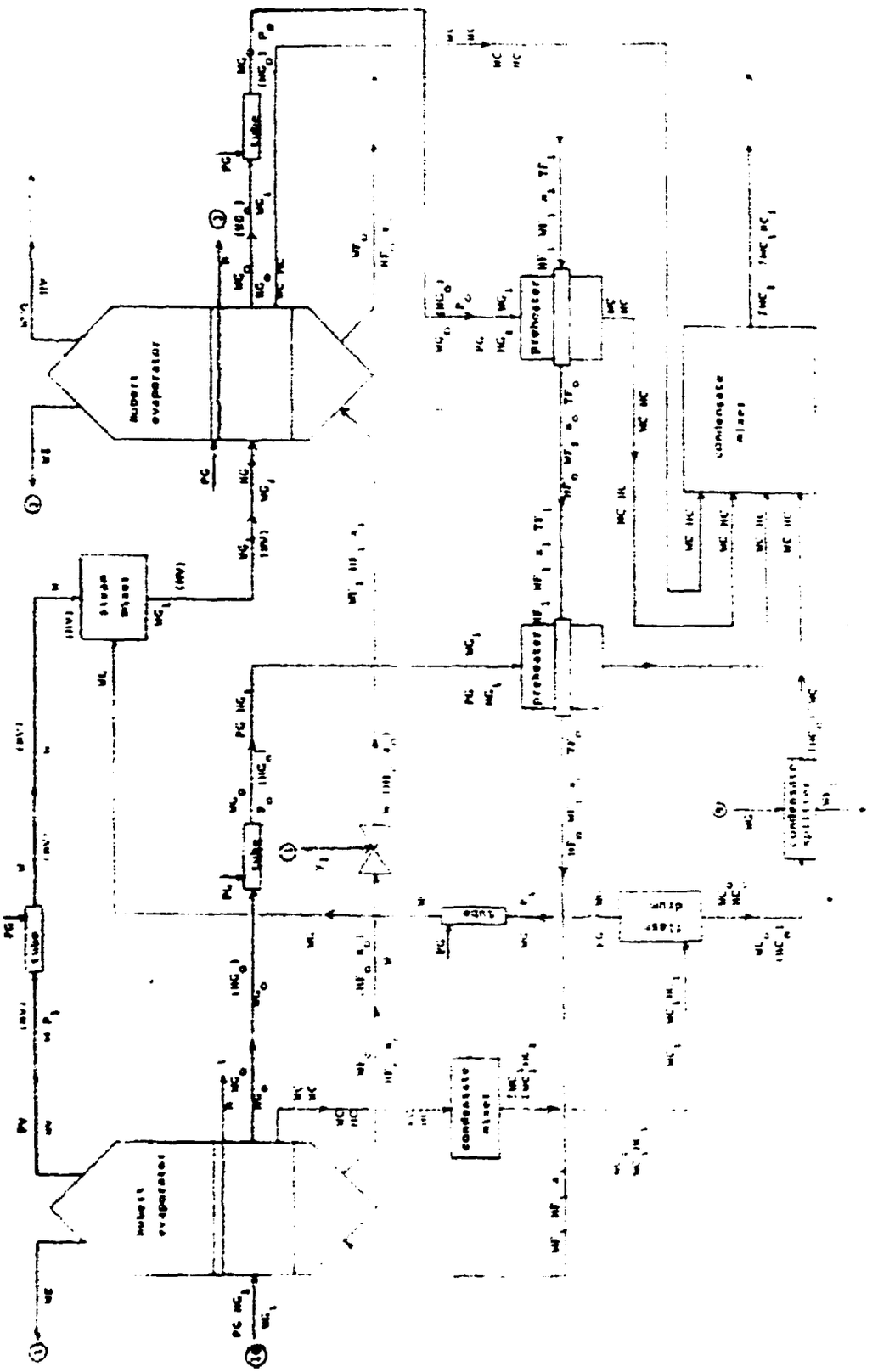


Fig. 5.19. Connections of schematic descriptions of submodels for a part of the multiple effect evaporator with dynamic calculation of the pressure.

6. CONCLUSION

In this chapter the results of the project are reviewed. The ways to programme and perform a simulation with and without the precompiler system are compared, and also how the user is relieved of some of the error prone work will be discussed. Implementation of the modular concept has an impact on programming, validation and modification of models and the advantages and disadvantages of this implementation will be commented on.

There will be a discussion of what is missing because of limitations due to the concept and due to lack of time. The precompiler system together with the runtime executive DYSIM86 and auxiliary programmes for pre- and post-processing are compared with some known simulation packages.

Finally the prediction of the future development of this and other simulation packages is sought. This will contain some of the ideas of the continued work on short- and long-term basis in our group which may include other types of simulation and usage of simulation in other areas.

6.1. Assessment of the results

There are two ways in which to assess the results of this project. First, the simulation package, which includes the precompiler system, the runtime executive DYSIM86 and auxiliary programmes, will be compared with what it was before the precompiler system was developed. Secondly, the simulation package is compared with other packages.

6.1.1 DYSIM now and then

The simulation package DYSIM consisted 'in the beginning' of only the runtime executive DYSIM86 which uses an input file that is modularly structured. The typical way of conducting a simulation was then to make some FORTRAN77 files with modules

and a connecting system, compile and link them to the runtime executive. In order to perform the simulation an input file with initial conditions and runtime commands had to be prepared as well as a list file with the symbolic names of the variables and the order with which they appear in the modules.

The list file provides the user with the possibility of using symbolic names for the variables when a reference is needed in the input file, during simulation and for post-processing. The runtime executive uses the list file to convert positions in internal arrays to symbolic names (or vice versa) and therefore there has to be the agreement in order of variables that was mentioned above. During testing of a module new variables may be added and care had to be taken to update the list file accordingly, and omissions were often the cause of errors. In Appendix L the FORTRAN77 code produced by the precompiler system is listed and the following statements can be supported by referring to the listing:

The transfer of variables from the connecting system to the modules is (normally) done by means of common blocks. Which ever way the variables are transferred the ordering of the variables and the names and sizes of the common block must be the same in the connecting system as in the module. In the case of adding a variable this was also not always done leading to errors. These errors could be hard to find since the integration was allowed to start, but the results were surprising which could lead to the suspicion that the new variable was causing the problem by improper modelling when it really was only an omission.

Apart from being able to model a plant the user has to be a FORTRAN77 programmer. The idea of a library with submodels for components could of course still be used, but instead more complicated programmes were made to model a module. This reduces the readability of the programme.

When the runtime executive initializes and performs accuracy control the output variables are transferred back and forth

between the arrays used by the runtime executive and by the modules and connecting system. This has absolutely nothing to do with modelling, but nevertheless it is necessary to deal with it. In order to do this efficiently the user had to acquire a rather detailed knowledge about how the runtime executive works. This is of course yet another place where the programme has to be updated when the number of variables in a module is being changed.

The transfer of variables from the runtime executive to the connecting system and (typically) onwards to the modules is done with common blocks. The way the variables have to be moved back and forth between common blocks in the connecting system to accomplish this transfer is made most conveniently by using arrays instead of the symbolic names that were used in the modules. Here the user has to have a list file to be able to convert between elements in arrays and symbolic names. Actually this list file was the "ancestor" to the list file being used by the runtime executive.

One more demand on the user for being able to use the simulation system as it were is that the user has to remember to make an entry point or a subroutine called by the runtime executive when the simulation is finished. This will typically be the place where a "profile" of the model is written, i.e. some of the variable values are printed. This entry point or subroutine and the connecting system are called by the runtime executive and the names and the syntax must be known.

Lastly it must be mentioned that there existed an auxiliary program for producing a list file. Also there exists an auxiliary program for making an input file and there is one for making plots and another for making tables giving the result of a simulation. The fact that they have been developed after the precompiler system, except for the mainframe plotting program, is not relevant in this discussion.

Now the simulation package DYSIM includes the precompiler system and the way to conduct a simulation is thus different. It

will be assumed that a library with thermodynamic and other functions and with submodules has been prepared. Now no FORTRAN77 files have to be made by the user. Instead, some source files to the precompiler system must be prepared, one for the connecting system and one for each module. These have to be processed by the precompiler system and those resulting files that contain FORTRAN77 code have to be compiled and linked to the runtime executive. The simulation can then be performed after an input file has been prepared.

At the top of the source files the input and output variables of the module or of the connecting system are declared using symbolic names. These names constitute the necessary information to make a list file making it possible to perform a simulation. This list file is made automatically by the precompiler system and this will ensure that any changes of a module will cause a correct updating of the list file.

The list file is made when the connecting system is being processed because at this time all the information about the system is known. At this time also the connections to and from the modules are known and the precompiler system almost completely relieves the user from thinking about this. The only connections that the user have to be concerned about are the connections to the input variables of the modules. This will typically be either just a transfer of an output variable from another module, i.e. a connection between modules, or an expression with connecting system parameters, i.e. an introduction of a disturbance. These connections are made using FORTRAN77-like statements.

If the submodule library is adequately well equipped the user will have to know almost nothing about the programing language FORTRAN77. When programing the modules all the components can then be modelled by using submodules and the only FORTRAN77 statements that should be written are the connections in the connecting system as described above. Here even the disturbances which may be a little complicated may be written using some functions in the library. Therefore it is right to say

that the user can perform simulation without knowing anything about FORTRAN77.

When the precompiler system is processing the connecting system the common blocks are automatically written in a consistent manner and the transfer of variables at initialization and accuracy control are also written in the correct syntax and most efficiently.

As stated earlier the variables are parts of arrays in the connecting system, but the precompiler system allows the user to write the symbolic names instead. This greatly increases the readability and therefore many errors are avoided.

It is not necessary for the user to remember to write the entry point that the runtime executive is calling before terminating a simulation. This entry point and another that is necessary if the user has used a special feature of DYSIM are automatically made a part of the connecting system.

If the simulation is performed on a PC the FORTRAN77 files that have been produced by the precompiler system must be compiled individually, and afterwards they are going to be linked to the compiled version of the runtime executive. This is most easily done using a "link file" that provides the linker with the necessary information. A link file is made by the precompiler system. If the simulation is performed on the mainframe the FORTRAN77 files are going to be compiled together with the runtime executive FORTRAN77 files and the most convenient way to do that is to make a file which uses some compiler instructions that includes the FORTRAN77 files. This file is made by the precompiler system.

In order to be able to use the precompiler system the user must know which syntax to use. If it is possible to avoid using FORTRAN77 statements only the syntax of the precompiler system has to be learned. This syntax is quite simple, especially compared to that of FORTRAN77. Of course, it is not necessary to be acquainted with all of the specialties of FORTRAN77, but it

is not really possible to say what subset of the FORTRAN77 language will suffice, so most of the language must be learned.

In Table 6.1 the major differences between DYSIM now, with the precompiler system, and then are listed.

	THEN	NOW
Usage of submodels	User's responsibility	Supported by the precompiler system
Usage of symbolic names in the connecting system	Not possible	Possible
Transfer of variables at initialization and at accuracy control	User's responsibility	Automatically
Preparation of a consistent set of common blocks	User's responsibility	Automatically
Preparation of list file	By the user	Automatically
Preparation of link file	By the user	Automatically

Table 6.1. The major differences between DYSIM without and with the precompiler system.

6.1.2. Pros and cons of modularity

The modular concept of DYSIM86 has been extended with one more level in the concept of the precompiler system as has been described earlier. The two levels are

1) usage of submodules when constructing a module, and
2) usage of modules when constructing a model of the total process plant.

The modular concept is seen in other simulation packages but mostly at the first of our two levels: usage of models of components. This allows reuse of models, or rather submodels, at several places in a model of a process plant.

The development of new ordinary programming languages all supports the concept of "structural programming" and a part of this concept is modularity. By modularity is not only meant usage of routines at several places in a programme, but equally important also the usage of routines for well-defined delimited purposes in the main programme. The reason for this is the possibility of individual testing of smaller units of the programme which of course is not as hard as having to consider every part of the programme if an error has found its way into the programme. This line of thought also applies for simulation programmes.

It is an advantage to use submodules from a library when modelling a plant because ultimately it is possible to construct a model without having to make one mathematical model oneself. This is the case if the library contains all the submodels one can wish. Also, if a submodule is to be replaced by another model of the same component that is either more or less detailed it is easily done merely by changing the library and then reprocessing the model. In some cases it may be necessary only to process the connecting system since it is only then that the submodules from the library are being picked out.

On the second level of modularity it is also possible to use a more or less detailed model. In both of the examples in this

report feedwater was extracted from a process plant and the feedwater was evaporated and steam fed to the process plant. In both of the examples the boilers were not modelled, in fact there are no modules for them. If a more thorough study of the two process plants was necessary it would be possible to make a model of the boiler in a module and simply add it to the other module(s) since as input variables they already have those variables that the boiler module more realistically is calculating. The thought of reusing modules is not as obvious as for submodules and therefore it is not felt to be a limitation that process plant specific parameters are part of the modules.

The mathematical model of a process plant consists of a mixed system of coupled ordinary nonlinear differential equations and algebraic equations. Together these equations express the relationships between the variables at any time. In other words they are all simultaneously true statements and as such should be solved simultaneously. In practice this will always mean the application of iteration, which of course is time-consuming.

If iteration is found to be too time-consuming one can settle on an approximation where the equations are not solved simultaneously, but old values for some of the variables are used. This is the case when there is an algebraic loop which is a series of algebraic equations that cannot be written in a sequence where each algebraic variable is calculated before it is being used. By sorting the statements the number of these kinds of variables should be reduced to a minimum.

The way that the precompiler system uses modules and submodules, i.e. using FORTRAN77 CALL statements, makes it impossible to sort statements across the two-level interfaces that are introduced by the two-level modular approach. This problem is also recognized in other simulation packages (CROSBIE et al, 1985).

Working with two levels gives a better overview of a model that is of moderate size but it can be felt as an unnecessary complication if the model is small. The submodules can presently be used only in the modules, so the model must be programmed using both levels if the submodule library is to be used.

6.2. What is missing?

First let us specify what it is that is "missing something". Another way to put it is to say that it must be specified on which time scale the "missing things" should be expected to be implemented if so desired. It was chosen to focus here on the "things that are missing" in DYSIM and which can be implemented within one year. Next it is stressed that it is not necessarily so that everything here is going to be implemented. The experiences gained from working with DYSIM will be used for development of a more user friendly system with modern techniques.

There are two versions of DYSIM, one for IBM PC or compatibles and one for Risø's Burroughs mainframe computer. The precompiler system on the two different types of computers are very much alike, the only differences relate to file names and the size of the systems that can be simulated. The runtime executive DYSIM86 on the two computers are dissimilar. The more or less batch-oriented Burroughs version was modified to the interactive PC version which allows an integration stop at any time, monitoring values of output variables, change of parameters, continuation of simulation, etc. and which works with menus and different windows each being scrolled individually. This is presently the status.

When the PC version of the runtime executive was made two restrictions were placed on it. Firstly, the size of the models that can be simulated was reduced by lowering the number of variables that are accessible to the user. Secondly, the choice of integration routines were reduced to the two Runge-Kutta methods with step size control (2.nd and 4.th order). On the mainframe version a library, ODEPACK (BYRNE and HINDMARSH, 1975), is available with implicit multistep algorithms. It is necessary to be able to choose implicit integration routines on the PC also.

During simulation of a transient the user may want to know whether a new steady state has been reached to decide to stop or go on with the integration. The most efficient way to do

this is to plot the time history so far for one or more of the output variables. This plotting facility during a stop in a transient calculation is desirable.

When a parameter study must be made the same transients must be calculated over and over again only varying a parameter. It would be nice if it was not necessary to have to stop the simulation programme to start a new transient with a new value for the parameter. It is already possible to change parameters at any time so it would be desirable to be able to run a transient, monitor the result, change a parameter and, as the new feature, to reinitialize and restart the programme.

As was discussed in 6.1.2 "Pros and cons of modularity" sorting of statements in a two-level modular approach is not straightforward. If the statements were sorted and the programme halted due to a runtime error reported by the FORTRAN77 runtime system the user would have a hard time finding out where in his own source file the problem arose. There are two answers to this and each of them are desirable features of its own right.

One way to handle the situation above would be to implement a routine in the precompiler system that suggests to the user a sequence to write the statements or MACRO commands. This would also detect algebraic loops which are potentially the cause of numerical instabilities.

Also it would be nice to have a routine that will interpret the FORTRAN77 runtime error and report the trouble by referring to the user's source file instead of the FORTRAN77 file created by the precompiler system.

On a longer term basis use of graphics for input of models and for debugging can be considered. These issues will be commented on later.

6.3. Comparison with some other simulation systems

When comparing the simulation system DYSIM with CSSLs it must be noticed that DYSIM does not include a simulation language as such while a CSSL (Continuous System Simulation Language) does. Most CSSLs of today are developed from the CSSL67 proposals (STRAUSS (editor), 1967) which includes a touch of "old-fashioned" analogue simulation. Forgetting this for the moment the main points of the proposals are

1) source files for a compiler are used for describing the model

2) the runtime executive controls the independent variable (time), takes care of initialization, etc.

3) the runtime executive terminates the simulation by applying decisions based on some logics being part of the model

4) there must be a programme to document the results of a simulation run

5) it must be possible to perform parameter studies

6) statements are going to be sorted

7) there must be means to indicate that a section of statements are not going to be sorted.

Sorting of statements is the only thing that is not supported presently by DYSIM.

One of the most widely used CSSLs is ACSL (MITCHELL and GAUTHIER, 1986) and it meets all of the CSSL67 proposals and a few details have been clarified as for example the introduction of the DISCRETE section for discrete events. In ACSL the DISCRETE section is executed when an event has occurred and here the user programmes the actions to take at this event. ACSL's runtime execu-

tive automatically adjusts the time step to "accurately" hit the time of the event that has been programmed. To do the same in DYSIM a submodule is used to adjust the time step and the user will programme the actions to be taken. ACSL has an INITIAL and TERMINAL section that are used for programming, among other things, the experiments to perform while one has to manually perform the experiments in DYSIM. ACSL has a sorting routine which may be considered an advantage as the other features above definitely are.

Among the drawbacks are the facts that the choice of integration parameters can be made part of the programme and that the initial conditions must be part of the programme. We strongly feel that the programme shall have only the values for those parameters of the model that are truly constant. It is recognized that in ACSL these parameters can be changed when starting the simulation but it does not seem to be a practical way to work. For small models it seems that ACSL may look more interactively controlled but for large systems one can fear that the overview is easily lost. The usage of the input file in DYSIM makes it easy for the user to specify initial conditions and choosing integration routine and plot variables and still it is possible to interactively change parameters during a simulation.

It is possible to use several integration routines in ACSL and this includes routines suited for stiff systems. It is even possible to use different integration routines on different parts of the model, but it is strongly advised against this not be done (MITCHELL and GAUTHIER, 1986) except for the advanced simulationist in special situations. Today stiff integration routines in DYSIM can be used only in the mainframe version and the same routine is applied for all of the model. This is not the whole truth since it is possible with the PC version of DYSIM to communicate with another PC at given "communication intervals" and it is therefore possible to perform parallel integration with all the implications that is introduced by that.

Both ACSL and DYSIM allow the usage of libraries but because of ACSL's sorting routine the statements with, for example, submodels are inserted each time the submodel is being used. The submodels can use other submodels in ACSL, while this is prohibited in DYSIM where a strict two-level modular approach is being applied.

The general application of simulation systems seems to be sacrificed when greater ease and more elaborate features are implemented. This is illustrated below for the case where graphical input is used when a model is being set up.

The first example is the Modular Modeling System, MMS, which can be used for simulation of power plants. The simulation system actually executing the user's model is ACSL but the input and analysis of the results are done by using the pre- and post-processors of MMS (SCHAAK, BARTELLS and LONG, 1986).

Another example is DIVA (HOLL, MARQUARDT and GILLES, 1987) which is well suited for chemical plants including columns. The graphical input is not yet implemented (spring 1987) but graphics are being used for showing the results of the simulation during execution of the simulation.

SPEEDUP (PERKINS and SARGENT, 1982) is an example of a simulation system that has been built for analysis of chemical plants. It applies the flowsheet concept which is wide spread in steady-state simulation systems and uses it also for dynamic simulation. This is an example of a simulation system that also applies two-level approach in the same sense as and for the same reasons that has been stated for DYSIM with precompilers. Sparse matrix techniques are being used for solving simultaneously the system of differential and algebraic equations. This is a huge simulation system that is available only on mainframes.

Before we proceed with proposals for future developments we shall conclude the assessment of the precompiler system interface to simulation with DYSIM86 by repeating the major points scattered in the report.

First of all it is clear that simulation of process plants using DYSIM86 is greatly simplified by using the precompiler system as an interface. The precompiler system manages most of the error prone programming as well as it provides the user with the possibility of avoiding the usage of FORTRAN77 statements (almost completely).

The runtime executive alone as well as the precompiler system applies the modular concept which is rather new in connection with simulation systems as compared to ordinary programming languages. The modular concept as it is seen DYSIM with precompiler system is only applied in a few other simulation systems or languages.

The modular concept introduced by the application of submodules taken from a library is shared with more simulation languages. This concept makes it possible to use a well-tested submodel. Very few other simulation languages are known to apply both levels of modularity.

DYSIM with the precompiler system separates the model of the process plant from the data determining the simulation experiments to be performed on the model as it is proposed in the CSSL81 proposal. It seems that the only points where DYSIM with the precompiler system does not comply with the CSSL81 proposal are the points concerning sorting of statements and segmentation that allows parallel processing.

The experiences we have had with the simulation system all indicate a greater reliability in the process of performing simulation experiments. We think that we have a modern and flexible simulation system that meets our present needs and since we have developed it ourselves, we shall be able to modify it to meet new needs as they emerge.

6.4. Proposals for future developments

In this section some proposals for future development in our simulation group are being presented. It is based on our own experiences, by reading about the developments going on elsewhere and listening to suggestions from users of simulation systems. These proposals are long-term based and as for the short-term based proposals in 6.2 they are not necessarily all going to be worked out. Also some of the proposals of other workers will be discussed.

When working with any programme, and therefore also with simulation systems, there are (at least) two different types of considerations. There is the power of the system and there is the user interface. The system should be powerful, i.e. give the user extensive service which the user does not even have to know. The user interface is part of the service that the user will see and it includes ways of working with the system, the help that can be offered, explanation of detected errors, etc.

Among the powers of the simulation system that the user should not be too concerned about is the choice of integration method. There has been done a lot of work on solving ordinary differential equations and there exists excellent routines to use, for example ODEPACK (BYRNE and HINDMARSH, 1975) that is being used on the Burroughs mainframe version of DYSIM.

In the general situation though the model consists of a mixture of differential and algebraic equations and this type of problem is not solved as easily. Actually all the equations should be solved simultaneously, but since this is rather time-consuming this is not always done. This problem can be solved by using the sequential modular approach, for example, known from steady-state simulation packages (JØRGENSEN, 1985), in dynamic simulation (HILLESTADT and HERTZBERG, 1986) as it has been done in DIVA (HOLL, MARQUARDT and GILLES, 1987). Here extrapolation of values for input variables are used as it is also suggested in future developments of other simulation systems (SKLAVER, 1986).

Another example (GANI, 1987) avoids the problem of extrapolation by tearing streams as it is done in steady state simulation packages. When time delays are present in a recycle loop interpolation is used. This is possible by restricting the choice of integration steps. Algebraic equations are calculated separately from differential equations. Here iterations are not necessary because algebraic variables are not allowed to depend on other algebraic variables. This restriction will not work, in general. The modular approach of the reference is used for reducing of the size of the Jacoby matrix by integrating each module separately, but with the same time step for every module. This is time saving for the part of calculation of the matrix but some modules may be integrated by time steps that are much smaller than necessary considering stability.

The ideas of using reduced Jacobians and interpolation of values for input variables should be studied in order to get a more efficient integration method. When assessing integration methods the benefits, i.e. higher accuracy, reduction of computer time consumption, increase stability, etc. must be compared with the drawbacks such as choosing parameters carefully which in the end may demand that the user be familiar with integration techniques.

It is seen in all of the references above concerning integration as well as those where others (CROSBIE, 1986) risk predictions of future simulation systems that the modular concept is considered to offer several advantages. The concept has been seen as well to call for some problems to be solved (think of, for example, sorting of statements). Therefore the modularity, which furthermore seems natural, must also be applied in the future, but the problems must be addressed. These problems are thought to be so complex for the moment that the user may have to be concerned with this when a model is being programmed. The sensitivity of the efficiency of the "correct" choice of modularization has been demonstrated (GONZALES et al, 1984).

The user interface includes setting up the model, executing the simulation, post-processing to evaluate the results of the simu-

lation and any help offered by the simulation system at any stage. The user interface cannot be discussed before it has been decided what kind of user one is interested in or rather what one expects from the user. It is today recognized that the user should be a domain expert and not necessarily a computer expert. As far as possible the user must be able to execute a simulation without knowing anything else about the computer other than how to turn it on (and off!). The user must be a domain expert by which it is meant that the system which is going to be simulated must be well known by the user so that the simulation can be evaluated in a reasonable manner.

So, the user is not expected to be too familiar with computers and computer or simulation languages. The simulation system must instead be a pleasant environment leading to the concept of Continuous System Simulation Environment, CSSE (LUKER, 1986). The environment is established by a set of coherent programmes which share and exchange information. There is nothing new in this but the development is expected to lead to more coherent, more self-explanatory, etc. simulation systems.

Ordinary programming languages and most simulation languages are based on a series of statements being executed sequentially. The trends today lead to systems where it is possible to write the programme just as the model builder wrote the model on the paper. The equations stating conservation of mass, energy, momentum, etc. are written directly and the simulation system will sort the equations and isolate the necessary variables, as for example SPEEDUP (PERKINS and SARGENT, 1982). If inconsistencies occur a warning is given before the model is executed. SPEEDUP applies the concept of streams which is necessary for using graphical input as SPEEDUP also is going to use.

Graphical input is already possible in some simulation systems but as discussed earlier this is only done in systems with a limited application area. The stream concept should be studied and used to connect submodels from a library using graphics and for example a "mouse". In this way it may be possible for users that are unfamiliar with computers to generate a model of a

process plant if the library is adequately well equipped with submodels to meet the needs. On the other hand it may greatly impede the construction of new submodels to the library which reduces the degree of openness of the library and thereby of the simulation system.

In simulators used for training the results of the simulation are shown during execution of the simulation, and the programme runs in real time. This may also be desirable in simulation systems that are not used for training. This is seen in DIVA (HOLL, MARQUARDT and GILLES,1987) which serves a delimited area of applications and therefore can be made in a way that excellently matches the problem. An example is the evolution of concentration or temperature profiles in columns. It should be studied whether a general scheme can be developed to show the course of the simulation graphically. One way to do this is to complicate the extension of the submodel library further by including graphical directions for this purpose.

All simulation systems allow graphical presentation of the results after termination of the simulation. Today no radical improvements in this are foreseen. Movie-making where the course of the simulation is shown in real time may be needed in special situations, but this rather looks like something that is connected to a simulator.

An area which is surely going to have an impact on simulation in the future is the usage of databases. Databases are now practicable on even PCs and on mainframes they are accessible at central data banks. Databases can be used for storage of all the (important) results of the simulation experiments (BIRTA and SO, 1986) where they can be examined later. Databases in data banks can be used for extracting the latest version of a given thermodynamic function and exchanging results and submodels all of which will relieve users in their work.

Some of the present simulation languages of which ESL is an example (CROSBIE et al, 1985) offer an interpreter that makes it possible to avoid compilations and linking to a runtime

executive whereby the user quickly can switch from model modification to simulation tests. The speed of the computers are increasing steadily so this line of development will probably not be beneficial even on shorter time horizons. The use of interpreters also makes it impossible to use routines written in other languages, as for example FORTRAN. This possibility is essential (WOOD, 1986) since so much work has been done earlier which must be available in the future also.

In any of the directions to be followed in future developments "debugging" aids must be considered. The debugging should begin already at the time of setting up the model. This is part of the idea of the envrironment in CSSEs (LUKER, 1986). At any step: pre-processing, compilation, execution, etc., the cause of errors must be clearly indicated and when the simulation system shows where the error was detected it must be made with reference to something that looks like what the user gave the simulation system as input.

For many years steady state simulation packages have been used for optimization of, for example, chemical process plants. The simulation system SPEEDUP (PERKINS and SARGENT,1982) which runs on mainframes has this facility, but with a few modifications the user can go on to execute a dynamic simulation. This possibility is desirable and a steady state finder should be developed.

The rapid propagation of expert systems opens new application areas for simulation and this will also have an impact on the construction of simulation environments where expertise can be acquired from the system.

The areas mentioned here represent many men working for many years and unfortunately we have to restrict our efforts to some of the topics only. Today and also in the future all of the areas are of interest and should be followed at least at a distance because some of the limitations that has been introduced along the way may be removed as new computers and new software will be marketed.

REFERENCES

ANSI (1978). American National Standard, programming language FORTRAN. American National Standards Institute.

BIRTA, L.G. and KHADR, A. (1979). A CSSL interface to GASP IV. In: Proceedings of the Summer Computer Simulation Conference held by the Society for Computer Simulation in Toronto, Canada, July 16-18 1979. pp 20-31.

BIRTA, L.G. and SO, M (1986). A database system for management of numerical experiments. In: Proceedings of the 1986 Summer Computer Simulation Conference held by the Society for Computer Simulation in Reno, Nevada, July 28-30, 1986. pp 38-43.

BOLMSTEDT, U. (1977). Simulation of the steady-state and dynamic behaviour of multiple effect evaporation plants. Part 2: dynamic behaviour. Computer Aided Design, vol. 9, no. 1, January 1977. pp 29-40.

BROWNE, J.C., DUTTON, J.E. and NEUSE, D.M. (1986). Introduction to graphical programming of simulation models. In: Proceedi of the 1986 Summer Computer Simulation Conference held by the Society for Computer Simulation in Reno, Nevada, July 2 30, 1986. pp 32-37.

BYRNE, G.D. and HINDMARSH, A.C. (1975). A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations. ACM Transactions on Mathematical Software, Vol. 1., No. 1, March 1975, pp 71-96.

CHRISTENSEN, P. la Cour (1974). Description of the Real Time PWR Power Plant Model PWR-PLASIM. Ris' Report No.318. 75 p.

CHRISTENSEN, P. la Cour (1979). Description of the Power Plant Model BWR-PLASIM Outlined for the Barseback 2 Plant. Risø-M-2190.

CHRISTENSEN, P. la Cour (1981). Description of a Simultion System DYSIM for Continuous Dynamic Processes. Risø-M-2271. 27 p.

CHRISTENSEN, P. la Cour (1986). Personal communication.

CHRISTENSEN, P. la Cour, KOFOED, J.E. and LARSEN, N. (1986). DYSIM - A Modular Simulation System for Continuous Dynamic Processes. Risø-M-2607. 72 p.

CHURCH, E.F. (1950). Steam Turbines. Third edition. McGraw-Hill Book Company, Inc.

COUGHANOWR, D.R. and KOPPEL, L.B. (1985). Process Systems Analysis and Control. 19. printing. McGraw-Hill Inc.

CROSBIE, R. (1986). Developments in ESL and other CSSL's for the 1990's. In: Proceedings of the 1986 Summer Computer Simulation Conference held by the Society for Computer Simulation in Reno, Nevada, July 28-30, 1986. pp 313-314.

CROSBIE, R.E. and HAY, J.E. (1982). Towards new standards for Continuous-System Simulation Languages. In: Proceedings of the 1982 Summer Computer Simulation Conference held by the Society for Computer Simulation in Denver, Colorado, July 19-21, 1982. pp 186-190.

CROSBIE, R.E., JAVEY, S., HAY, J.L. and PEARCE, J.G. (1985). ESL - A new Continuous System Simulation Language. SIMULATION 44:5, pp 242-246.

DDS (1985). Om sukker og sukkerfabrikation hos De Danske Sukkerfabrikker. Aktieselskabet De Danske Sukkerfabrikker.

ELMQVIST, H. (1975). SIMNON An interactive simulation programme for nonlinear systems. USER'S MANUAL. Report 7502. Department of Automatic Control. Lund institute of Technology.

ELMQVIST, H., ÅSTRÖM, K.J. and SCHÖNTHAL, T. (1986). SIMNON - User's Guide for MS-DOS Computers. Department of Automatic Control, Lund Institute of Technology.

GANI, R. (1987). A dynamic simulation package for chemical processes. In: Proceedings on the 29'th Annual Meeting of the Scandinavian Simulation Society. pp 166-180.

GONZALES, S., MENDEZ, E., KUHLMAN, F. and CASTELAZO, I. (1984). Large scale power plant model development. Part I: Modularization. In: Proceedings of the International conference on power plant simulation held by Instituto do investigaciones electricas, November 19-21, Cuernavaca, Morelos, Mexico. pp 359-366.

HANSEN, L.A. and SØLTOFT, P. (1980). Kemiske enhedsoperationer. 4. edition. Akademisk forlag. Universitetsforlaget København.

HILLESTADT, M. and HERTZBERG, T. (1986). Dynamic simulation of chemical engineering systems by the sequential modular approach. Computers & Chemical Engineering, Vol. 10, No. 4, pp 377-388.

HOLL, P., MARQUARDT, W. and GILLES, E.D. (1987). DIVA - A powerful tool for dynamic process simulation. Presented at the Meeting on Use of Computers in Chemical Engineering (CEF 87) held by the European Federation of Chemical Engineering at Taormina, Sicily, Italy, April 26-30, 1987.

JØRGENSEN, L.B. (1985). Et fleksibel datamaskineprogram til beregning af masse- og energibalancerne i en sukkerfabrik. Erhvervsforskerprojekt EF98.

JØRGENSEN, L.B. (1986). Personal communication.

KOFOED, J.E. (1986). Dynamic simulation of a power plant. RP-7-86. 105 p.

KOFOED, J.E. (1987). The proceedings of the 1987 Summer Computer Simulation Conference July 27-30, 1987, Montreal, Quebec, Canada (1021 pages) pp. 9-13.

LAM, C.F. (1978). A pre-compiler for a GPSS processor. In: Modeling and simulation. 9th annnual Pitsburgh conference, 1978, vol. 9, no. 1-4. pp 837-841.

LUKER, P.A. (1986). From CSSL to CSSE. In: Languages for continuous system simulation. Proceedings of the Conference on Continuous System Simulation Languages held by the Society for Computer Simulation in San Diego, California, January 23-25 1986. pp 53-56.

MÄKELÄ, M. (1981). Mathematisches Formulieren und digitales Simulieren einer Verdampfstation in der Rübenzuckerindustrie. Zuckerind. 106 (1981) Nr. 11. pp 989-993.

MITCHELL and GAUTHIER, Assoc. (1986). Advanced Continuous Simulation Language. Reference Manual.

PERKINS, J.D. and SARGENT, R.W.H (1982). SPEEDUP: A computer program for steady state and dynamic simulation and design of chemical processes. Computer-aided process design and analysis. AIChE symposium series.

RIMVALL, M. and CELLIER, F (1986). Evolution and perspectives of simulation languages following the CSSL standard. Modeling, identification and control, 1986, vol.6, no.4, pp 181-199.

SCHAAK, R.D., BARTELLS, P.S. and LONG, A.B. (1986). Development of a PC version of the modular modeling system using the ACSL simulation language and the MMS-EASE+ pre and post processor. In: Languages for continuous system simulation.

Proceedings of the Conference on Continuous System Simulation Languages held by the Society for Computer Simulation in San Diego, California, January 23-25 1986. pp 7-12.

SCHLECHTENDAHL, E.G. (1970). A Dynamic System Simulator for Continuous And Discrete Changes of State. KFK 1209. 46 p.

SKLAVER, E.R. (1986). Dynamic simulation in 1991: An Exxon viewpoint. In: Proceedings of the 1986 Summer Computer Simulation Conference held by the Society for Computer Simulation in Reno, Nevada, July 28-30, 1986. pp 292-296.

SMITH, J.M. and VAN NESS, H.C. (1975). Introduction to Chemical Engineering Thermodynamics, Third edition, McGraw-Hill Book Company, New York.

STRAUSS, J.C. (editor), AUGUSTIN, D.C., FINEBERG,M.S., JOHNSON, B.B., LINEBARGER, R.N. and SANSOM, F.J. (1967). Ths SCi Continuous System Simulation Language (CSSL). SIMULATION December 1967. pp 281-303.

WOOD, R.K. (1986). Experience with existing languages - What's missing. In: The Proceedings of the 1986 Summer Computer Simulation Conference held by the Socierty for Computer Simulation in Feno, Nevada, July 28-30, 1986. pp 306-309.

ZEIGLER, B.P. (1976). Theory of Modeling and Simulation. John Wiley and Sons, New York.

APPENDIX A

DESCRIPTION OF SUBMODULES USED FOR SIMULATION OF POWER PLANTS

In this appendix the submodules that are part of the library which was used in the simulation of a part of a power plant are described. These descriptions are revised versions of those appearing in another work (KOFOED, 1986). There has been found an error in the model of the PI-controller which would occur only in situations where the output of the controller had to be limited. In the simulations in this project the system were never driven to such conditions so this error does not affect the results of the simulations.

1. TIME DELAY

1.1. Name

TIDEL

1.2. Description

A property of a flow is being delayed in time.

1.3. Characteristics

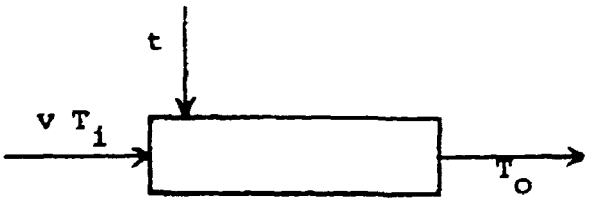
This model was built to simulate the situation of a time-varying flow of water through a tube. The temperature (for example) of the water is varying. The time history is recorded and it is

assumed that the water in the tube does not mix, i.e. it is moving like a stick. Initially the tube is assumed to be filled with water at the initial temperature. Then it is possible at any time to calculate the amount of water leaving the tube and, which is the purpose of the submodule, the temperature of the water that is leaving the tube.

1.4. Prerequisites

The time history is stored in some arrays that are managed by the runtime executive DYSIM86. Two functions in DYSIM86 must be available. The arrays are rather big and therefore it has been chosen to leave them out of DYSIM86 by default and it is the user's responsibility to include them in the simulation programme. This is done by including the function DELAY.

1.5. Symbols



1.6. Identifiers

Name	Type	Dimension	Description
x	STV	(e.g. m)	Total amount of flow that has entered
v	INP	(e.g. m/s)	Flow rate
t	INP	s	Time
T_i	INP	(e.g. C)	Attribute of flow being delayed
T_o	ALG	(e.g. C)	Value of delayed variable
L	PAR	(e.g. m)	Size of delaying device

1.7. Model

This submodule can be used for simulation of the time delay of temperature fluctuations in a flow of water passing a tube but any time delay that can be transformed to a similar description can also be modelled with this submodule.

The time delay is simulated by assuming that a water front flows undisturbed down the tube. The time at which this front leaves the tube is equal to the time it takes to fill the tube after the front has entered the tube. Let the linear flow rate be v and let the length of the tube be L. The position of the front entering initially in an infinitely long tube is described simply by

$$\dot{x} = v$$

The water that is leaving the tube is located at the position x-L. A function, TRNSTM, which is part of the function DELAY, now gives the time delay.

$$\tau = \text{TRNSTM('TAU',x,x-L)}$$

in units of timesteps. The string 'TAU' will be commented on shortly. The outlet temperature is then calculated by using another function in DELAY

$$T_o = \text{DEADTM('TOUT',}T_i\text{,}\tau\text{)}$$

which also passes the inlet temperature to DYSIM86 for storage and later retrieval.

The inlet temperatures are stored by DYSIM86 and there must of course be one storage for each time delay. Each storage must be uniquely identified and this is done using the strings in the two functions above. These strings are constructed by the pre-compiler system from the name of the module that is calling the submodule. This is done using the UNI command in the source file for the submodule.

2. STEAM COOLER, TYPE A AND B

2.1. Names

SCA and SCB

2.2. Description

A streaming fluid is heated by condensing steam.

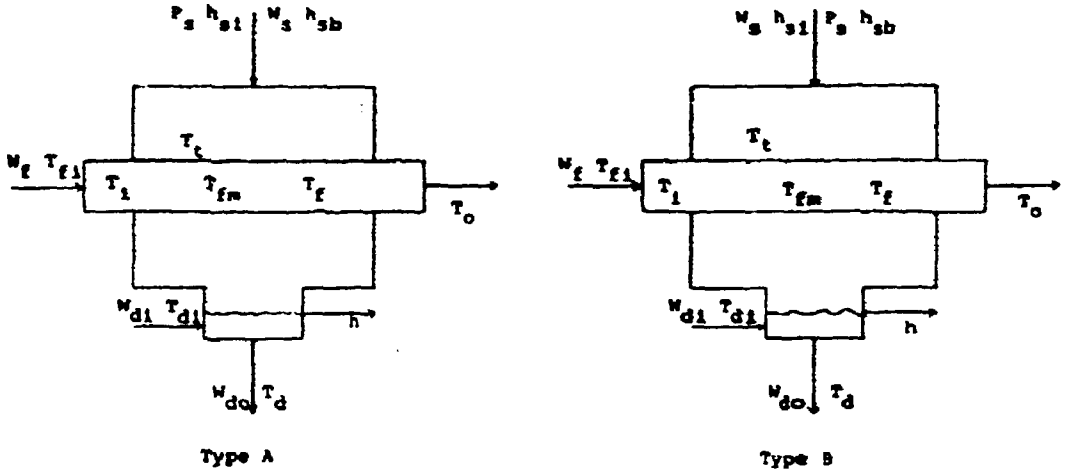
2.3. Characteristics

The steam is saturated. A point model is used. The mean temperature of the fluid is calculated ignoring the tube. The mean temperature governs the heat transfer from the steam through the tube to the fluid. The fluid enters the steam cooler in an inlet chamber and leaves through an outlet chamber. The drainage is mixed with a flow of another drainage in a tank. The level of the tank is measured. The drainage leaves the tank with a certain flow rate.

2.4. Prerequisites

Functions TMA, HFS, HGS, and TSAT must be available.

2.5. Symbols



2.5. Identifiers

Name	Type	Dimension	Description
P_s	A:INP	bar	Pressure of steam
P_s	B:STV,ext	---	-------- -- -----
W_s	A:STV,ext	kg/s	Steam flow into cooler
W_s	B:INP	----	--- ---- ---- --------
h_{si}	INP	MJ/kg	Specific enthalpy of steam flow into cooler
h_{sb}	ALG,ext	----	do, out of cooler
W_f	INP	kg/s	Flow of fluid
T_{fi}	INP	^oC	Inlet temperature of fluid
T_i	STV,int	--	Fluid temp. in inlet chamber
T_f	STV,int	--	----- ----- -- tube
T_o	STV,ext	--	----- ----- -- outlet chamber
T_{fm}	ALG,int	--	----- ----- -- tube, mean
T_t	STV,int	--	Tube temp.
T_{di}	INP	--	Inlet temp. of drainage
W_{di}	INP	kg/s	----- flow -- --------
W_{do}	INP	----	Outlet ---- -- --------
T_{do}	STV,ext	^oC	------ temp. -- --------
h	STV	m	Height of drainage in tank

k_{st}	PAR	MW/K	Heat transf. coeff. outside tube
k_{tf}	PAR	----	Heat transf. coeff. inside tube
C_t	PAR	MJ/K	Heat capacity of tube
C_i	PAR	----	---- -------- -- fluid, inlet
C_o	PAR	----	---- -------- -- -----, outlet
C_f	PAR	----	---- -------- -- -----, tube
c_{pf}	PAR	MJ/kg K	Spec. heat capacity of fluid
c_{pd}	PAR	-------	----- ---- -------- -- drainage
ρ_d	PAR	kg/m^3	Density of drainage
A_d	PAR	m^2	Tank area
V	B:PAR	m^3	Gas volume. NB: only type B
ρ_g'	B:PAR	kg/m^3bar	$\partial\rho_g/\partial P_s$

2.7. Model

Using conservation of energy the net flow of heat into the inlet chamber $W_f c_{pf}(T_{fi}-T_i)$ is equal to the heat $C_i\dot{T}_i$ accumulated in the inlet chamber at the temperature rate $\dot{T}_i$. The same procedure is used for the outlet chamber giving

$$\dot{T}_i = W_f c_{pf}(T_{fi}-T_i)/C_i \qquad \text{(SC1)}$$

$$\dot{T}_o = W_f c_{pf}(T_f -T_o)/C_o \qquad \text{(SC2)}$$

It is seen that a point model is used.

For the heat transfer a point model is also used. Therefore a mean temperature of the fluid in the tube is needed. A weight factor B is calculated (see Appendix F). The total heat transfer coefficient kL is needed (L is the length of the cooler)

$$kL = \left(\frac{1}{k_{st}} + \frac{1}{k_{tf}}\right)^{-1} \qquad \text{(SC3)}$$

as well as the auxiliary variable H

$$H = W_f c_{pf} \tag{SC4}$$

where H is the heat flow per degree through the tubes.

Then B is calculated using the function TMA

$$B = B(kL/H) = TMA(kL,H) \tag{SC5}$$

The mean temperature of the fluid in the tube is the weighted sum of the inlet and outlet temperature

$$T_{fm} = B \cdot T_i + (1-B) \cdot T_f \tag{SC6}$$

The steam is assumed to be saturated and therefore the steam temperature T_s can be determined by the saturation temperature at pressure P_s

$$T_s = T_{sat}(P_s) \tag{SC7}$$

where T_{sat} is a thermodynamic function that must be available. Now the heat transfer to and from the tube can be calculated.

$$Q_{st} = k_{st}(T_s-T_t) \tag{SC8}$$

$$Q_{tf} = k_{tf}(T_t-T_{fm}) \tag{SC9}$$

where Q_{st} is the heat flow from the steam to the tube and Q_{tf} is the flow from the tube to the fluid. The net flow of heat $(Q_{st}-Q_{tf})$ to the tube causes a temperature change at rate

$$\dot{T}_t = (Q_{st}-Q_{tf})/C_t \tag{SC10}$$

see (SC1-2). Energy conservation applied on the fluid yields

$$\dot{T}_f = [W_f c_{pf}(T_i-T_f)+Q_{tf}]/C_f \tag{SC11}$$

The drainage may be boiling due to heating by the drainage

flowing into the tank from the outside. The amount of drainage W_c that is evaporated, if superheating is present, is assumed to be proportional to the amount M_d of drainage and the square of the degree of superheating (T_d-T_s). The proportionality constant chosen is 0.008.

$$W_e = \begin{cases} 0 & \text{if } T_d<T_s \\ 0.008 \cdot M_d(T_d-T_s)^2 & \text{if } T_d>T_s \end{cases} \qquad \text{(SC12)}$$

$$M_d = hA_d\rho_d \qquad \text{(SC13)}$$

The amount of energy Q_e used to evaporate the steam is

$$Q_e = W_e(h_{gs}-h_{fs}) \qquad \text{(SC14)}$$

where h_{gs} and h_{fs} are the specific enthalpy of the steam and the drainage at saturation. These must be available as the thermodynamic functions named HGS(T) and HFS(T).

It is assumed that the net flow of steam entering the cooler is all condensing

$$W_c = W_s+W_e \qquad \text{(SC15)}$$

Stating conservation of mass on the drainage in the tank you get

$$\dot{h} = (W_c+W_{di}-W_{do}-W_e)/(A_d\rho_d) \qquad \text{(SC16)}$$

Energy conservation is applied to yield the rate of change of drainage temperature T_d (see Fig. A.1). It is assumed that the specific heat capacity c_{pd} of the drainage is constant, and furthermore that the zero point of the thermodynamic functions are at t = 0°C, i.e. that HFS(0) = 0 or h_{fs} = 0 at t = 0°C, and therefore the following relation holds true

$$h_{fs} = c_{pd}T \qquad \text{(SC17)}$$

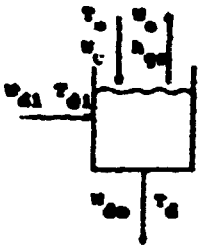


Fig. A.1. The tank.

The amount of energy in the condensate is $c_{pd}M_dT_d$. Application of energy conservation yields

$$\frac{d}{d\tau}(c_{pd}M_dT_d) = c_{pd}W_{di}T_{di}+c_{pd}W_cT_s-c_{pd}W_{do}T_d-W_eh_{gs} \qquad (SC18)$$

$$\frac{d}{d\tau}(c_{pd}M_dT_d) = c_{pd}(\dot{M}_dT_d+M_d\dot{T}_d) \qquad (SC19)$$

M_d is equal to $hA_d\rho_d$ (see SC16). This is multiplied by $c_{pd}T_d$ and subtracted on both sides of (SC18) after using (SC19). You get

$$c_{pd}M_d\dot{T}_d = c_{pd}[W_{di}(T_{di}-T_d)+W_c(T_s-T_d)]-W_e(h_{gs}-c_{pd}T_d) \qquad (SC20)$$

Since $c_{pd}T_d=h_{fs}$ the last term is $W_e(h_{gs}-h_{fs})$ which is Q_e (see SC14). Inserting and isolating you get

$$\dot{T}_d = [W_{di}(T_{di}-T_d)+W_c(T_s-T_d)-Q_e/c_{pd}]/M_d \qquad (SC21)$$

The way the steam flow W_s is handled makes the difference between the two types of steam coolers. Type A is characterized by the fact that the steam flow is determined by the amount of heat transferred from the steam to the tubes, while the steam flow is an input variable in type B. In type A energy conservation determines W_s while in type B mass conservation determines the steam pressure in the gas cooler.

Consider first type A. The net flow of energy Q_s transferred into the steam volume is

$$Q_s = Q_{st} - Q_e \qquad \text{(SC22)}$$

where the total amount Q_{st} needed for heat transfer to the tube is reduced by the amount Q_e released when the steam evaporated from the drainage, is condensed on the tubes. If $W_s>0$ there is a net flow into the steam cooler and the specific enthalpy h_g of the steam is h_{si}. In this case the backward flow specific enthalpy h_{sb} is zero. If $W_s<0$ you have backward flow, and the specific enthalpy h_g of the steam as well as the backward flow specific enthalpy h_{sb} is h_{gs}. In short, the cases are described by

$$h_g = \begin{cases} h_{si} & \text{if } W_s>0 \quad \text{(forward flow)} \\ h_{gs} & \text{if } W_s<0 \quad \text{(backward flow)} \end{cases} \qquad \text{(SC23)}$$

The assumption that all the steam entering the volume is condensed determines the necessary flow Z of condensing steam according to the following statement of energy conservation

$$Q_s = Z(h_g - h_{fs}) \qquad \text{(SC24)}$$

If the needed flow Z differs from the actual flow W_s the inlet steam flow must be increased. This is approximated by a first-order equation. In the case of forward flow you get

$$\dot{W}_s = (Z-W_s)/\tau \qquad \text{(forward flow)} \qquad \text{(SC25)}$$

The time constant τ reflects the acceleration time for the steam. The time constant τ must be kept small compared with the thermal dynamics of the heater, but in order to get a reasonable timestep the value should not be chosen too small.

In special cases more steam is generated in the tank than is condensed at the tubes, i.e. $W_e>Z$. For this to be consistent with the assumption of all steam entering the volume is con-

densed a backward flow is established. This is also modelled by a first-order equation. The difference between the backward flow W_S ($W_S<0$) and the needed backward flow $Z-W_e$ yields

$$\dot{W}_S = [(Z-W_e)-W_S]/\tau \qquad \text{(backward flow)} \qquad \text{(SC26)}$$

In the steam cooler <u>type B</u> conservation of mass determines the pressure in the steam cooler. The inlet flow of steam W_S (input variable) as well as the condensation rate W_C (SC15) and evaporation rate W_e (SC12) is known. Mass conservation yields

$$\dot{M} = W_S+W_e-W_C \qquad \text{(SC27)}$$

where $M = \rho_g V$ is the mass of the steam.

$$\dot{M} = \frac{d}{d\tau}(\rho_g V) = V\frac{d\rho_g}{d\tau} = V\frac{\partial\rho_g}{\partial P_S}\frac{dP_S}{d\tau} = V\rho_g'\dot{P}_S \qquad \text{(SC28)}$$

introducing the notation ρ_g' for $\partial\rho_g/\partial P_S$. Now the differential equation describing P_S can be written.

$$\dot{P}_S = (W_S+W_e-W_C)/V\ \rho_g' \qquad \text{(SC29)}$$

3. HEAT EXCHANGER

3.1. Name

HX

3.2. Description

Heat is transferred from one flow to another. This causes a temperature decrease in one flow and temperature increase in the other.

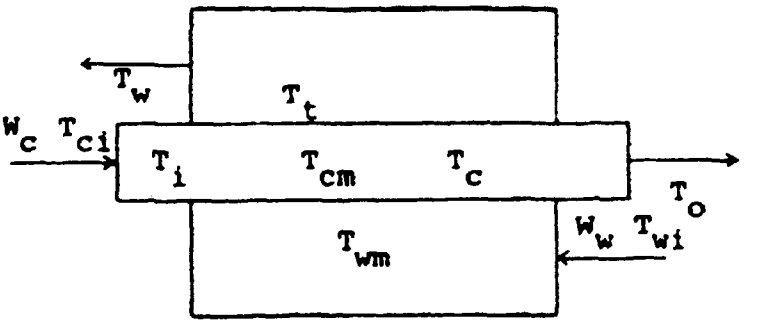
3.3. Characteristics

The two flow directions are opposite, i.e. it is a countercurrent heat exchanger. A point model is used as an approximation for the distributed heat exchanger. The mean temperatures of the flows are calculated and used for heat transfer calculation. The flow inside the tube enters the heat exchanger in an inlet chamber and leaves through an outlet chamber.

3.4. Prerequisites

Function TMA must be available.

3.5. Symbols



3.6. Identifiers

Name	Type	Dimension	Description
T_i	STV,int	^oC	Fluid temperature in inlet chamber
T_o	STV,ext	^oC	- - outlet -
T_c	STV,int	^oC	- - at outlet from tube
T_w	STV,ext	^oC	- - - - volume outside tube
T_t	STV,int	^oC	Tube temperature

T_{ci}	INP	^oC	Fluid temperature, entering inlet camber
T_{wi}	INP	^oC	- - - volume outside tube
W_c	INP	^oC	Flow of fluid in chambers and in tubes
W_w	INP	kg/s	- - - entering volume outside tube
T_{cm}	ALG,int	^oC	Mean temperature of fluid in tube
T_{wm}	ALG,ext	^oC	- - - - outside tube
k_{ct}	PAR	MW/K	Heat transfer coefficient inside tube
k_{wt}	PAR	MW/K	- - - outside tube
C_t	PAR	MJ/K	Heat capacity of tube
C_c	PAR	MJ/K	- - - fluid inside tube
C_w	PAR	MJ/K	- - - - outside -
C_{ci}	PAR	MJ/K	- - - - in inlet chamber
C_{co}	PAR	MJ/K	- - - - - outlet -
c_{pc}	PAR	MJ/K	Specific heat capacity of fluid in tube
c_{pw}	PAR	MJ/K	- - - - - - outside tube

3.7. Model

The net energy flow into the inlet chamber $W_c c_{pc}(T_{ci}-T_i)$ is equal to the accumulation of heat $\dot{T}_i C_{ci}$ in the inlet chamber. Applying conservation of energy gives

$$\dot{T}_i = W_c c_{pc}(T_{ci}-T_i)/C_{ci} \qquad \text{(HX1)}$$

A point model has been used assuming a well-stirred tank. The same procedure for the outlet chamber yields

$$\dot{T}_o = W_c c_{pc}(T_c-T_o)/C_{co} \qquad \text{(HX2)}$$

Now attention is turned to the point model of the ordinary heat exchanger. A mean temperature of the fluid in the tube is needed. A weight factor B is calculated (see Appendix F). The total heat transfer coefficient kL is needed (L is the length of the heat exchanger)

$$kL = \left(\frac{1}{k_{st}} + \frac{1}{k_{tf}}\right)^{-1} \qquad \text{(HX3)}$$

The auxiliary variable

$$H = \frac{W_w c_{pw} \cdot W_c c_{pc}}{W_w c_{pw} - W_c c_{pc}} \qquad \text{(HX4)}$$

is ensured to be limited even though the denominater approaches zero. Using kL and H a weight factor B is calculated (see Appendix F)

$$B = B(kL/H) = TMA(kL,H) \qquad \text{(HX5)}$$

where TMA is a real function that must be available. The mean temperatures can now be calculated

$$T_{cm} = B \cdot T_i + (1-B) \cdot T_c \qquad \text{(HX6)}$$

$$T_{wm} = (1-B) \cdot T_{wi} + B \cdot T_w \qquad \text{(HX7)}$$

The heat flow from outside the tube to the tube is

$$Q_{wt} = k_{wt}(T_{wm} - T_t) \qquad \text{(HX8)}$$

and from the tube to the fluid inside the tube

$$Q_{tc} = k_{tc}(T_t - T_{cm}) \qquad \text{(HX9)}$$

By stating energy conservation around the tube the rate of change of tube temperature $\dot{T}_t$ can be calculated. The net flow of energy to the tube $(Q_{wt}-Q_{tc})$ is set equal to the accumulated heat $(\dot{T}_t C_t)$. Isolating $\dot{T}_t$ you get

$$\dot{T}_t = (Q_{wt} - Q_{tc})/C_t \qquad \text{(HX10)}$$

Energy conservation applied to the two fluids yield

$$\dot{T}_c = [W_c c_{pc}(T_i - T_c) + Q_{tc}]/C_c \qquad \text{(HX11)}$$

$$\dot{T}_w = [W_w c_{pw}(T_{wi} - T_w) - Q_{wt}]/C_w \qquad \text{(HX12)}$$

4. MIXER

4.1. Name

MIX

4.2. Description

Two flows of fluids are entering a volume where they mix and heat or cool the content. A flow, equal to the sum of the two incoming flows, leaves the volume.

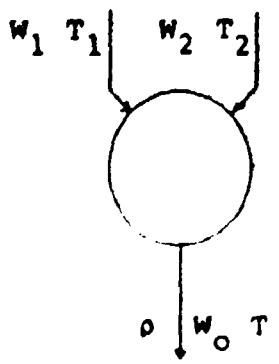
4.3. Characteristics

A point model of the volume (tank) is used leading to a first order system. The fluid is considered incompressible.

4.4. Prerequisites

None.

4.5. Symbols



4.6. Identifiers

Name	Type	Dimension	Description
T	STV,ext	^oC	Temperature of fluid in tank and of outlet flow
W_o	ALG,ext	kg/s	Outlet flow of fluid
W_1	INP	kg/s	Inlet flow 1
W_2	INP	kg/s	- - 2
T_1	INP	^oC	Temperature of inlet flow 1
T_2	INP	^oC	- - - - 2
ρ	INP	kg/m^3	Density of fluid in tank
V	PAR	m^3	Volume of tank

4.7. Model

Since the fluid is considered incompressible the outlet flow W_o is equal to the sum of the two inlet flows

$$W_o = W_1+W_2 \qquad \text{(MIX1)}$$

Stating energy conservation you get

$$\dot{T} = (W_1T_1+W_2T_2-W_oT)/V\rho \qquad \text{(MIX2)}$$

by assuming that the specific heat capacity of the three flows are equal.

5. PI CONTROLLER

5.1. Name

PI

5.2. Description

Output is proportional to the sum of the error and the errors integrated, where the error is the difference between the input and the set point.

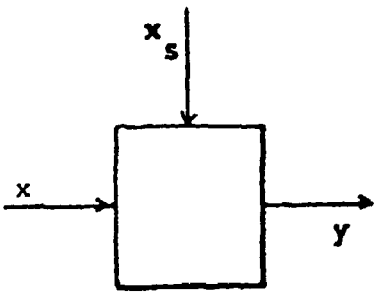
5.3. Characteristics

Input and set point are limited to the measuring range of the measuring device. The output of the controller is limited to the unit range: 0-1. If the output is in either limit and the error will push the output beyond the limit, the rate of change of output is set to zero. Besides the amplification in the standard controller, the proportional band, a gain factor is used for the output. It is the output of the composite controller that is limited to the unit range.

5.4. Prerequisites

The limiting functions ALIM and ALIMD.

5.4. Symbols



5.6. Identifiers

Name	Type	Dimension	Description
x	INP	anything	Input signal
x_s	INP	-	Set point
x_o	PAR	-	Lower limit in measuring range
x_m	PAR	-	Upper - - - -
z	STV,int		Integral of error
P_b	PAR	anything	Proportional band. Notice dim.
τ_I	PAR	s	Reset time
G	PAR		Gain. Standard controller: G=1
y	ALG,ext		Output signal

5.7. Model

The signal x and the set point x_s are limited to the measuring range x_o to x_m.

The error e is given by

$$e = x - x_s \qquad \text{(PI1)}$$

where x and x_s are in the measuring range. The error is divided by the proportional band P_b. Note that P_b is the error required to change the output from just closed to just fully open. This is contrary to the usual terminology where P_b is given as the percentage of the measuring range (COUGHANOWR and KOPPEL, 1985).

This is simply written

$$\varepsilon = \frac{e}{P_b} \qquad \text{(PI2)}$$

ε is the normalized error and z is the integral of this divided by the reset time τ_I

$$\dot{z} = \frac{\varepsilon}{\tau_I} \tag{PI3}$$

z is limited to the interval $[0, 1/G]$ (see below) and if z is in either limit and $\dot{z}$ has a sign that would force z outside the interval then $\dot{z}$ is set equal to zero. For example, if $z = 1/G$ and $\dot{z} > 0$ then the value of $\dot{z}$ is changed to 0 keeping z at 1/G. The output of a standard controller is $z + \varepsilon$, but in this composite controller this sum is multiplied by the gain G

$$y = G(z + \varepsilon) \tag{PI4}$$

The output y is limited to the unit interval.

APPENDIX B

LIBRARY WITH FUNCTIONS AND SUBMODULES
USED FOR SIMULATION OF POWER PLANTS

This appendix contains a listing of the library that was used for the simulation of a part of a power plant. This includes the functions that are specially designed for the submodules, some "water functions" (thermodynamic functions) and the submodules. The submodule PI that models the PI-controller is the one that was used in the simulations but as it was described in Appendix A it is in error. This error never had any influence on the results. For a correct, standard PI-controller use the submodule in the other library, Appendix G.

```
C *
C
C     FUNCTIONS
C     =========
C
      FUNCTION ALIM(XMIN,X,XMAX)
C
C        CAN BE MADE AN INTERNAL FUNCTION.
C
      REAL ALIM,X,XMAX,XMIN
C
      ALIM=AMAX1(XMIN,AMIN1(X,XMAX))
      RETURN
      END
CCCCCC      ======

      FUNCTION ALIMD(Y,XMIN,X,XMAX)

C
      REAL ALIMD,Y,XMIN,X,XMAX,Z
C
```

```
      IF((X.GT.XMIN) .AND. (X.LT.XMAX)) THEN
        Z=Y
      ELSE IF(((X.LE.XMIN) .AND. (Y.LT.0.)) .OR.
     £         ((X.GE.XMAX) .AND. (Y.GT.0.))) THEN
        Z=0.
      ELSE
        Z=Y
      END IF
      ALIMD=Z
C
      RETURN
      END
CCCCCC      ======
      FUNCTION HFS(T)
      HFS=(((.208812E-10*T-.532844E-8)*T+.898397E-6)*T
     £ +.412051E-2)*T+.135802E-2
      RETURN
      END
CCCCCC      ======
      FUNCTION HGS(T)
      HGS=(((-.300545E-10*T+.262449E-8)*T-.171730E-5)*T
     £ +.193629E-2)*T+2.49979
      RETURN
      END
CCCCCC      ======
      FUNCTION TMA(X,W)
*     ABSOLUTE PRECISSION BETTER THAN 1.E-6
      REAL X,W,E,Z,B
      IF(ABS(W).GT.1.E-10*X) THEN
        Z=X/W
      ELSE
        Z=SIGN(1.E-10,W)
      END IF
      IF((1.E-3.LE.ABS(Z)) .AND. (ABS(Z).LE.1.E2)) THEN
        B=1./Z+1./(1.-EXP(Z))
      ELSE IF(ABS(Z).LE.1.E-3) THEN
        B=.5-Z/12.
      ELSE IF(1.E2.LE.Z) THEN
        B=1./Z
```

```
      ELSE IF(Z.LE.-1.E2) THEN
        B=1.+1./Z
      END IF
      TMA=B
      RETURN
      END
CCCCCC      ======
      FUNCTION TSAT(P)
      IF(P.GT.11.) GOTO 3
      IF(P.GT.1.1) GOTO 2
1     TSAT=((((21.0001*P-124.324)*P+281.872)*P-315.384)*P+207.750)*P
     £ +28.5272
      RETURN
2     TSAT=((((.142502E-2*P-.579261E-1)*P+.918869)*P-7.38848)*P
     £ +36.7123)*P+69.7477
      RETURN
3     TSAT=((((-.435533E-9*P+.193481E-6)*P-.348376E-4)*P+.329490E-2)*
     £ -.180916)*P+6.82706)*P+126.888
      RETURN
      END
CCCCCC      ======
&
&
&     ============================================================
&
&                    NOW  THE SUBMODULES  FOLLOW
&
&     ============================================================
&
&
!     SUBMO      HX
&     =====
&                HEAT EXCHANGER
&
!FUN                 TMA;
!UNI
!STV                 TT ,TW ,TI ,TC ,TO ;
!INP                 WW,WC,
!                    TWI,TCI;
```

```
!PAR                KWT,KTC,CT,CW,CCI,CC,CCO,CPW,CPC;
!ALG                TWM,TCM;
&
&          TT       TEMPERATURE OF TUBE
&          TW       TEMPERATURE OF WARM FLUID, AT OUTLET AND IN HX
&          TI       TEMPERATURE OF COLD FLUID, IN INLET CHAMBER
&          TC       TEMPERATURE OF COLD FLUID, IN HX
&          TO       TEMPERATURE OF COLD FLUID, IN OUTLET CHAMBER
&
&          WW       FLOW OF WARM FLUID
&          WC       FLOW OF COLD FLUID
&          TWI      TEMPERATURE OF WARM FLUID AT INLET
&          TCI      TEMPERATURE OF COLD FLUID AT INLET
&
&          KWT      HEAT TRANSFER COEFFICIENT; WARM FLUID TO TUBES
&          KTC      HEAT TRANSFER COEFFICIENT; TUBES TO COLD FLUID
&          CT       HEAT CAPACITY OF TUBES
&          CW       HEAT CAPACITY OF WARM FLUID
&          CCI      HEAT CAPACITY OF COLD FLUID IN INLET CHAMBER
&          CC       HEAT CAPACITY OF COLD FLUID IN TUBES
&          CCO      HEAT CAPACITY OF COLD FLUID IN OUTLET CHAMBER
&          CPW      SPECIFIC HEAT CAPACITY OF WARM FLUID
&          CPC      SPECIFIC HEAT CAPACITY OF COLD FLUID
&
&          TWM      MEAN TEMPERATURE OF WARM FLUID.
&          TCM      MEAN TEMPERATURE OF COLD FLUID.
C *
      REAL KL,B,QWT,QTC,HW,HC,HWC
      REAL H,K
C *
      DATA K/1.E20/
C *   NON-VANISHING FLOWS ASSUMED IN THIS MODEL.
C *   TOTAL HEAT TRANSFER COEFFICIENTS FOR
C *   HEAT TRANSFER FROM WARM FLUID TO TUBES / TUBES TO COLD FLUID
      KL=1./(1./KWT+1./KTC)
      HW=WW*CPW
      HC=WC*CPC
      HWC=HW-HC
      IF(K*ABS(HWC).GT.ABS(HW*HC)) THEN
```

```
        H=HW*HC/HWC
      ELSE
        H=SIGN(K,HWC)*SIGN(1.,HW*HC)
      END IF
C *   USER MUST GUARANTEE THAT NEITHER IS ZERO : NON-VANISHING FLOWS
C *   WEIGHT-FACTOR
      B=TMA(KL,H)
C *   MEAN TEMPERATURE OF WARM FLUID / COLD FLUID
      TWM=(1.-B)*TWI + B*TW
      TCM=B*TI + (1.-B)*TC
C *   HEAT TRANSFER FROM WARM FLUID TO TUBES / TUBES TO COLD FLUID
      QWT=KWT*(TWM-TT)
      QTC=KTC*(TT-TCM)
C *   ENERGY BALANCE
      TT :(QWT-QTC)/CT
      TW :(WW*CPW*(TWI-TW)-QWT)/CW
      TI :WC*CPC*(TCI-TI)/CCI
      TC :(WC*CPC*(TI-TC)+QTC)/CC
      TO :WC*CPC*(TC-TO)/CCO
      RETURN
      END
&     ------------------------------------------------------------
*
!     SUBMO           MIX
&     =====
&                     TWO FLOWS ARE MIXED TO ONE
!STV       T;
!INP       W1,T1,W2,T2,RHO;
!PAR       V;
!ALG       WO;
*
      WO=W1+W2
      T :(W1*T1+W2*T2-WO*T)/(V*RHO)
      RETURN
      END
&
&     ------------------------------------------------------------
C *
!     SUBMO       PI
```

```
&      =====
&
&                  PI-REGULATOR
&
!FUN     ALIM,ALIMD;
!TXT     REPORT;
!STV     Z;
!INP     X ,XS;
!  PAR X0      ,XM   ,PB,TAU,G    ;
!ALG Y;
&            Z           INTEGRAL OF ERROR
&
&            X           INPUT SIGNAL
&            XS          SET POINT
&
&            X0,XM       RANGE OF INPUT
&            PB          PROPORTIONAL BAND
&            TAU         TIME CONSTANT
&            G           GAIN
&
&            Y           OUTPUT SIGNAL
C *
C *
      REAL SIGNAL,SETPNT,EPSLON
C *
C *
C *   PI-CONTROLLER
C *
      SIGNAL=ALIM(X0,X,XM)
      IF(SIGNAL.NE.X) REPORT='INPUT OUT OF MEASURING RANGE.'
      SETPNT=ALIM(X0,XS,XM)
      IF(SETPNT.NE.XS) REPORT(30:)='SETPOINT OUT OF MEASURING RANGE.'
      EPSLON=(SIGNAL-SETPNT)/PB
      Z=ALIM(0.,Z,1.)
      Z :ALIMD(EPSLON/TAU,0.,Z,1.)
      Y=ALIM(0.,G*(EPSLON+Z),1.)
      RETURN
      END
```

```
&
&     ------------------------------------------------------------
C *
!     SUBMO       SCA
&     =====
&
&                 STEAM COOLER (TYPE A) WITH A DRAIN TANK
&
!FUN                  HFS,HGS,TMA,TSAT;
!STV                  WS ,H ,TD ,TT ,TI ,TF ,TO ;
!INP                  PS,HSI,TDI,WDI,WDO,WF,TFI;
!PAR                  TAU,KST,KTF,CT,CI,CF,CO,CPF,CPD,RHOD,AD;
!ALG                  HSB,TFM;
&
&           WS        FLOW OF STEAM INTO STEAM COOLER
&           H         HEIGHT OF DRAINAGE IN TANK
&           TD        TEMPERATURE OF DRAINAGE
&           TT        TEMPERATURE OF TUBES
&           TI        TEMPERATURE OF FEEDWATER IN INLET CHAMBER
&           TF        TEMPERATURE OF FEEDWATER IN TUBES
&           TO        TEMPERATURE OF FEEDWATER IN OUTLET CHAMBER
&
&           PS        PRESSURE OF STEAM
&           HSI       ENTHALPI OF STEAM AT INLET
&           TDI       TEMPERATURE OF DRINAGE AT INLET
&           WDI       FLOW OF DRAINAGE AT INLET
&           WDO       FLOW OF DRAINAGE AT OUTLET
&           WF        FLOW OF FEEDWATER
&           TFI       TEMPERATURE OF FEEDWATER AT INLET
&
&           TAU       TIME CONSTANT FOR RATE OF CHANGE OF STEAM FLOW
&           KST       HEAT TRANSFER COEFFICIENT; STEAM TO TUBES
&           KTF       HEAT TRANSFER COEFFICIENT; TUBES TO FEEDWATER
&           CT        HEAT CAPACITY OF TUBES
&           CI        HEAT CAPACITY OF FEEDWATER IN INLET CHAMBER
&           CF        HEAT CAPACITY OF FEEDWATER IN STEAM COOLER
&           CO        HEAT CAPACITY OF FEEDWATER IN OUTLET CHAMBER
&           CPF       SPECIFIC HEAT CAPACITY OF FEEDWATER
&           CPD       SPECIFIC HEAT CAPACITY OF DRAINAGE
```

```
&           RHOD       DENSITY OF DRAINAGE
&           AD         AREA OF DRAINAGE TANK
&
&           HSB        ENTHALPI AF STEAM IN BACKWARD FLOW
&           TFM        MEAN TEMPERATURE OF FEEDWATER.
C *
      REAL WSP,HP,TS,MD,WE,HS,HSA,HD,QE,QST,QS,Z,WC,KL,B
C *
      TS=TSAT(PS)
      IF(H.LT.1.E-5) H=1.E-5
      MD=H*AD*RHOD
      IF(TD-TS.GT.0.) THEN
C *      SUPER-HEATING -> EVAPORATION AT RATE
         WE=.008*MD*(TD-TS)**2.
      ELSE
         WE=0.
      END IF
C *   ENTHALPY OF H2O AS STEAM AND AS WATER AT TEMPERATURE TSAT
      HS=HGS(TS)
      HD=HFS(TS)
      QE=WE*(HS-HD)
      QST=KST*(TS-TT)
      QS=QST-QE
      IF(WS.GE..0) THEN
C *   (NORMAL) FORWARD FLOW
        HSA=HSI
        HSB=0.
C *     Z IS THE NECASSARY INPUT FLOW OF STEAM
        Z=QS/(HSA-HD)
        WSP=(Z-WS)/TAU
      ELSE
C *   BACKWARD FLOW
        HSA=HS
        HSB=HSA
C *     Z IS THE NECESSARY "FLOW" OF CONDENSING STEAM
C *     THE NET PRODUCTION OF STEAM IS (WE-Z)
C *     THE FLOW OF STEAM IN(!) IS -(WE-Z)
        Z=QS/(HSA-HD)
```

```
      END IF
      WS :WSP
      WC=WS+WE
      HP=(WC+WDI-WDO-WE)/(AD*RHOD)
      IF((HP.LT.0.) .AND. (H.LE.1.E-5)) HP=0.
      H :HP
      TD :(WDI*(TDI-TD)+WC*(TS-TD)-QE/CPD)/MD
      TI :WF*CPF*(TFI-TI)/CI
      TO :WF*CPF*(TF-TO)/CO
      KL=1./(1./KST+1./KTF)
      B=TMA(KL,WF*CPF)
      TFM=B*TI+(1.-B)*TF
      QTF=KTF*(TT-TFM)
      TF :(QTP-WF*CPF*(TF-TI))/CF
      TT :(QST-QTF)/CT
C *
      RETURN
      END
&
&     ------------------------------------------------------------
C *
!     SUBMO       SCB
&     =====
&
&                 STEAM COOLER (TYPE B) WITH A DRAIN TANK
&
!FUN                  HFS,HGS,TMA,TSAT;
!STV                  PS ,H ,TD ,TT ,TI ,TF ,TO ;
!INP                  WS,HSI,TDI,WDI,WDO,WF,TFI;
!PAR                  KST,KTF,CT,CI,CF,CO,CPF,CPD,RHOD,AD,V,RG;
!ALG                  HSB,TFM;
&           PS        PRESSURE OF STEAM
&           H         HEIGHT OF DRAINAGE IN TANK
&           TD        TEMPERATURE OF DRAINAGE
&           TT        TEMPERATURE OF TUBES
&           TI        TEMPERATURE OF FEEDWATER IN INLET CHAMBER
&           TF        TEMPERATURE OF FEEDWATER IN TUBES
&           TO        TEMPERATURE OF FEEDWATER IN OUTLET CHAMBER
&
```

```
&           WS          FLOW OF STEAM INTO STEAM COOLER
&           HSI         ENTHALPI OF STEAM AT INLET
&           TDI         TEMPERATURE OF DRAINAGE AT INLET
&           WDI         FLOW OF DRAINAGE AT INLET
&           WDO         FLOW OF DRAINAGE AT OUTLET
&           WF          FLOW OF FEEDWATER
&           TFI         TEMPERATURE OF FEEDWATER AT INLET
&
&           KST         HEAT TRANSFER COEFFICIENT; STEAM TO TUBES
&           KTF         HEAT TRANSFER COEFFICIENT; TUBES TO FEEDWATER
&           CT          HEAT CAPACITY OF TUBES
&           CI          HEAT CAPACITY OF FEEDWATER IN INLET CHAMBER
&           CF          HEAT CAPACITY OF FEEDWATER IN STEAM COOLER
&           CO          HEAT CAPACITY OF FEEDWATER IN OUTLET CHAMBER
&           CPF         SPECIFIC HEAT CAPACITY OF FEEDWATER
&           CPD         SPECIFIC HEAT CAPACITY OF DRAINAGE
&           RHOD        DENSITY OF DRAINAGE
&           AD          AREA OF DRAINAGE TANK
&           V           STEAM VOLUME
&           RG          DIFF. QUOTIENT OF RHO WITH RESPECT TO PRESSURE
&
&           HSB         ENTHALPI AF STEAM IN BACKWARD FLOW
&           TFM         MEAN TEMPERATURE OF FEEDWATER.
C *
      REAL HP,TS,MD,WE,HS,HSA,HD,QE,QST,WC,KL,B
C *
      TS=TSAT(PS)
      IF(H.LT.1.E-5) H=1.E-5
      MD=H*AD*RHOD
      IF(TD-TS.GT.0.) THEN
C *      SUPER-HEATING -> EVAPORATION AT RATE
         WE=.008*MD*(TD-TS)**2.
      ELSE
         WE=0.
      END IF
C *   ENTHALPY OF H2O AS STEAM AND AS WATER AT TEMPERATURE TSAT
      HS=HGS(TS)
      HD=HFS(TS)
      QE=WE*(HS-HD)
```

```
      QST=KST*(TS-TT)
      IF(WS.GE..0) THEN
C *   (NORMAL) FORWARD FLOW
        HSA=HSI
        HSB=0.
      ELSE
C *   BACKWARD FLOW
        HSA=HS
        HSB=HSA
      END IF
      WC=(QST+WS*(HS-HSA))/(HS-HD)
      PS :(WS+WE-WC)/(V*RG)
      HP=(WC+WDI-WDO-WE)/(AD*RHOD)
      IF((HP.LT.0.) .AND. (H.LE.1.E-5)) HP=0.
      H :HP
      TD :(WDI*(TDI-TD)+WC*(TS-TD)-QE/CPD)/MD
      TI :WF*CPF*(TFI-TI)/CI
      TO :WF*CPF*(TF-TO)/CO
      KL=1./(1./KST+1./KTF)
      B=TMA(KL,WF*CPF)
      TFM=B*TI+(1.-B)*TF
      QTF=KTF*(TT-TFM)
      TF :(QTF-WF*CPF*(TF-TI))/CF
      TT :(QST-QTF)/CT
C *
      RETURN
      END
&
&     ------------------------------------------------------------
*
!     SUBMO          TIDEL
&     *****
&
&                    TIME DELAY
&
!UNI
!STV       X;
!INP       T,V,TI;
!PAR       L;
```

```
!ALG       TO;
&
&      Because of UNI-command a string variable 6 characters long
&         will be transferred. This will uniquely identify the
&         place in users programme where this subroutine was
&         called.
&            1) running character , start 'A'
&            2) name of submodule (TIDEL) , 5 characters long.
&
       REAL             TAU
       CHARACTER*7      TT,DT
*
       IF(T.LT.0.) THEN
*      STEADY STATE RUN
         XP=0.
       ELSE
         XP=V
       END IF
       X :XP
       TT='T'//IDENT
       DT='D'//IDENT
 1000  FORMAT(A1,I5.5)
       TAU=TRNETM(TT,X,X-L)
       TO=DEADTM(DT,TI,TAU)
*
       RETURN
       END
```

APPENDIX C

LISTING OF SOURCE FILES THAT ARE BEING USED FOR SIMULATION OF A PART OF A POWER PLANT

Here the source files to the precompiler system are listed. These source files were used for simulation of a part of a power plant. The simulation was performed at a time where only a preliminary version of the precompiler system had been developed and the source file with the present version of the precompiler system.

The use of text variables (strings) has been demonstrated but it is not safe to use them as it was done in this example.

The submodules that was used in this simulation are described in Appendix A and the library that was used is listed in Appendix B.

The list file, that was manually written because the preliminary version of the precompiler system did not include this facility, is listed in Appendix D and one of the input files that was used is listed in Appendix E.

Turbine Module, Listing

```
*
* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$$$$$$$$$$$$$$        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$$$$$                           $$$$$$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$          TURBINE SECTION          $$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$$$$$                           $$$$$$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$$$$$$$$$$$$$$        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*
```

```
C *
!     MODUL          TU
&
!FUN       HGS  ,HFS  ,TSAT ,TMA  ;
&
!STV       PV   ,PH   ,HHP  ,
!          PT   ,HRTV ,HRHV ,PM   ,TTR  ,
!          PL   ,HLP  ,
!          TCON ,TCO  ,MAC  ;
!INP       PE   ,  WS(5)    ,XV   ,
!          ALTV ,AMOV ,
&          INPUT VARIABLES TO LP-TURBINE : WS(1-3)  ,
!          WCON ,WCW  ,TCI  ,HSB(3)     ;
!ALG       WSL  ,WV   ,WH   ,WT   ,SH   ,HV   ,HH   ,XHO  ,
!          WTI  ,WTO  ,WRO  ,HTI  ,HTO  ,HRO  ,TRO  ,TRM  ,WM   ,WWW  ,
!          WL   ,WLO  ,SL   ,HC   ,
!          PC   ,TCM  ,
!          PS(5),HSI(5)     ,ET   ,EH   ,EL   ,WRH(2)     ,TRH(2)     ;
&
&
*
C *
*
      INTEGER        NP
      PARAMETER      (NP=14)
*
      REAL KE   ,KV   ,KH   ,VH   ,RGH  ,GH   ,
     &     KHL  ,KVM  ,VR   ,VT   ,VMO  ,RGR  ,CKQR ,KQT  ,CTR  ,
     &     KL   ,VL   ,RGL  ,GL   ,
     &     BC   ,KCW  ,CPC  ,CCW  ,CCT  ,
     &     PVR(3)    ,A(2) ,HPVR(3)    ,HPVR5
      REAL           AV,KLR,KQR,CPR,PR,HPSPH,HPSPT,HGSPT,SPSPT,SPSPC
      REAL           B,TM,HHO,TTO,XH,HM,HL,QCT,QCW,TE,TH,TT,TL,YV,YH,YL
      INTEGER        I,J
      REAL           VSL,RGV,QTR,QRR,H,E
*
*
      REAL           AVX(NP),AVY(NP)
```

```
C *   HP-VALVE CHARACTERISTICS
      DATA AVX/.000,.200,.250,.300,.350,.500,.550,
     &        .600,.650,.700,.750,.800,.900,1.00/
      DATA AVY/.000,.055,.110,.195,.295,.635,.740,
     &        .825,.883,.927,.955,.975,.990,1.00/
C *   STEAM LINE VOLUME AND D(RHO)/DP
      DATA VSL,RGV/100.,.54/
*          VSL = 79.08 -> 100 TO REDUCE DYNAMICS
C *
C *   ------------------------------------------------------------------

C *
C *   FLOW COEFFICIENTS
      DATA KE,KV,KH,KHL,KVM,KL /.3306671E-5,24.86,13.,1630.,150.,121.35/
C *   VOLUMES
      DATA VH,VR,VT,VMO,VL /10.,240.,284.,18.6,156./
C *   RHO DIFFERNTIATED WITH RESPECT TO PRESSURE
      DATA RGH,RGR,RGL /.51,.47,.425/
C *   TURBINE EFFICIENCIES
      DATA GH,GL /.781,.87867/
C *   HEAT TRANSFER COEFFICIENTS AND CALCULATION OF
      DATA KQT,KCW,CKQR /4.54,50.,294./
C *   HEAT CAPACITY, SPEC. HEAT CAP.
      DATA CTR,CCW,CCT,CPC /5.315,446.,73.,4.2E-3/
C *   EXTRACTION PRESSURE CALCULATIONS
      DATA PFVR,A /.0379,.171,.32,.1235,-1.225E-5/
C *   EXTRACTION ENTHALPY CALCULATIONS
      DATA HFVR,HFVR5 /.299,.563,.691,.293/
C *   MEAN TEMPERATURE CALCULATION
      DATA BC /.455/
C *
C *   ------------------------------------------------------------------
C *
*
      DATA   I/1/
*     ----------------------------------------------------------------
*
```

```
*     STATEMENT FUNCTIONS
*
      SGS(T)=(((.192112E-9*T-.223860E-6)*T+.967903E-4)*T
     £ -.254866E-1)*T+9.13896
      SPS(T)=(((-.119848E-10*T+.369736E-7)*T-.242318E-4)*T
     £ +.150920E-1)*T+.449447E-2
      SGSU(P,H)=(2.7487^-.0720765*P)*H
     £ +(-1.01691-.303783*P+.0452131*P**2)*H**2
      CPLP(P,T)=(((-.469811E-7*T+.345247E-5*P+.283806E-4)*T
     £ -.107016E-4*P*P-.195340E-2*P-.492290E-2)*T
     £ +.326455E-2*P*P+.292683*P+2.16220)/1000.
      PSAT(T)=((.649403E-6*T+.883179E-6)*T+.629256E-3)*T+.522211E-2
*
*     ------------------------------------------------------------
*     HIGH PRESSURE TURBINE WITH VALVE
*
*     FUNCTIONS
*       HGS,HPS,SGS,SPS
*
*     STATE VARIABLES
*       PV            STEAM PRESSURE BEFORE VALVE (NB REHEATER)
*       PH            STEAM PRESSURE IN TURBINE INLET CHAMBER
*       HEP           ENTHALPY OF WET STEAM  IN ----- ------
*
*     INPUT VARIABLES
*       PE            STEAM PRESSURE BEFORE STEAM LINE
*       PT            ----- -------- IN REHEATER
*       WM            ----- FLOW THROUGH VALVE : STEAM LINE TO REHEATER
*       WS(4-5)       ----- LOAD BY STEAM COOLER 4 AND 5
*       XV            VALVE OPENING
*
*     PARAMETERS
*       KE            FLOW COEFFICIENT FOR STEAM LINE
*       KV            ---- ---------- -- VALVE
*       KH            ---- ---------- -- TURBINE
*       VH            VOLUME OF TURBINE INLET CHAMBER
*       RGH           DIFFERENTIAL QUOTIENT OF RHO WITH RESP. TO PRESSURE
*       GH            EFFICIENCY OF TURBINE
*
```

```
*     ALGEBRAIC VARIABLES
*       WSL          STEAM FLOW IN STEAM LINE
*       WV           ----- FLOW THROUGH VALVE     (CBO)
*       WH           ----- ---- ------- TURBINE   (CBO)
*       WT           ----- ---- TO REHEATER       (NB REHEATER)
*       SH           ENTROPY OF STEAM IN INLET CHAMBER (CBO)
*       HV           SPECIFIC ENTHALPY OF STEAM FLOW INTO INLET CHAMBER
*       HH           -------- -------- -- ----- ---- OUT OF INLET CHAMB
*       HHO          -------- -------- -- ----- ---- --- -- TURBINE (NB
*       XHO            STEAM QUALITY    -- ----- ---- --- -- ------- ---
*
      TE=TSAT(PE)
      HV=.995*HGS(TE)+.005*HFS(TE)
*     AREA OF VALVE OPENING IS CALCULATED
  10  IF(AVX(I).LE.XV) THEN
        IF(I.LT.NP) THEN
          IF(AVX(I+1).GE.XV) THEN
            AV=(XV-AVX(I))/(AVX(I+1)-AVX(I)) *(AVY(I+1)-AVY(I)) + AVY(I
          ELSE
            I=I+1
            GOTO 10
          END IF
        ELSE
          AV=1.
        END IF
      ELSE
        IF(I.GT.1) THEN
          I=I-1
          GOTO 10
        ELSE
          AV=0.
        END IF
      END IF
      PR=PH/PV
      IF(PR.LT..577) THEN
        YV=1.
      ELSE
        YV=SQRT(1.-5.5888*((PR-.577)**2.))
      END IF
```

```
      PR=PT/PH
      IF(PR.LT..7) THEN
        YH=1.
      ELSE
        YH=1.-(100./9.)*((PR-.7)**2.)
      END IF
      WV=KV*PV*YV*AV
      WH=KH*PH*YH
      PH :(WV-WH)/(VH*RGH)
      HH=HHP/(VH*RGH*PH)
      HHP :WV*HV-WH*HH
      TH=TSAT(PH)
      TT=TSAT(PT)
      HFSPH=HFS(TH)
      XH=(HH-HFSPH)/(HGS(TH)-HFSPH)
      SH=XH*SGS(TH)+(1.-XH)*SFS(TH)
*     ASSUME GH=1.
      SFSPT=SFS(TT)
      HGSPT=HGS(TT)
      HFSPT=HFS(TT)
      XHO=(SH-SFSPT)/(SGS(TT)-SFSPT)
      HHO=XHO*HGSPT+(1.-XHO)*HFSPT
*     CORRECT FOR EFFICIENCY
      HHO=HH-GH*(HH-HHO)
      XHO=(HHO-HFSPT)/(HGSPT-HFSPT)
      WT=WH-(WS(4)+WS(5))
*
*     ------------------------------------------------------------
*
*     REHEATER WITH TWO VALVES
*
*     FUNCTIONS
*       HGS , HFS , TSAT
*
*     STATE VARIABLES
*       PT          STEAM PRESSURE IN REHEATER
*       HRTV        ENTHALPY OF STEAM BEFORE SUPER HEATER
*       HRHV        -------- -- -----   IN   ----- ------
*       PH          STEAM PRESSURE IN TUBES
```

```
*       TTR          TUBE TEMPERATURE
*
*     INPUT VARIABLES
*       WT           STEAM FLOW FROM TURBINE
*       XHO          ------ QUALITY OF STEAM FROM TURBINE
*       HHO          SPECIFIC ENTHALPY OF STEAM FROM TURBINE
*       PV           STEAM PRESSURE BEFORE VALVE
*       PL           ------ -------- IN CONDENSER
*       ALTV         VALVE OPENING : CONNECTION TO LOW PRESSURE TURBINE
*       AMOV         ------ -------- : ---------- -- STEAM LINE
*
*     PARAMETERS
*       RHL          FLOW COEFFICIENT FOR VALVE : CONNECTION LP-TURBINE
*       KVM          ---- ---------- --- ------ : ---------- STEAM LINE
*       VR           VOLUME OF SUPERHEATER CHAMBER
*       VT           ------- -- MOISTURE SEPARATION CHAMBER
*       VHO          ------- -- STEAM IN TUBES
*       RGR          DIFFERENTIAL QUOTIENT OF RHO WITH RESP. TO PRESSUR
*       CKQR         CONSTANT FOR OBTAINING HEAT TRANSFER COEFFICIENT
*       RQT          HEAT TRANSFER COEFFICIENT : CONDENSING STEAM - TUB
*       CTR          ---- CAPACITY OF TUBE
*
*     ALGEBRAIC VARIBLES
*       WTI          FLOW OF 'DRIED' STEAM  INTO  MOISTURE SEPARATOR(MS
*       WTO          ---- -- -------- ------ OUT OF --------- ----------
*       WRO          ---- -- SUPERHEATED STEAM OUT OF REHEATER
*       HTI          SPECIFIC ENTHALPY OF DRIED STEAM :  INTO  MS
*       HTO          --------- --------- -- ------ ------ : OUT OF --
*       HRO          --------- --------- -- SUPERHEATED STEAM AT OUTLET
*       TTO          TEMPERATURE OF STEAM OUT OF MS AND OF MOISTURE
*       TRO          ---------- -- ------ --- -- REHEATER
*       TRM          MEAN TEMPERATURE OF STEAM IN SUPERHEATER
*       TM           TEMPERATURE OF CONDENSED STEAM
*       WM           FLOW OF CONDENSING STEAM THROUGH VALVE
*       WMC          ---- -- CONDENSED  ------ OUT OF TUBE
*       WWM          ---- -- WATER REMOVED FROM WET STEAM
*
*     FURTHERMORE HGSPT AND HFSPT ARE USED
*
```

```
      WTI=(WRO/.995)*WT
      HTI=.995*HGSPT+.005*HFSPT
      WRO=WRL*(PT-PL)*ALTV
      WTO=(VR*WTI+VT*WRO)/(VT+VR)
      PT :(WTI-WRO)/(RGR*(VT+VR))
      HTO=HRTV/(VT*RGR*PT)
      HRO=HRRV/(VR*RGR*PT)
      IF(HTO.LE.HGSPT) THEN
        TTO=TT
      ELSE
        TTO=TT+(HTO-HGSPT)/CPR
      END IF
*       CPR          SPECIFIC HEAT CAPACITY OF STEAM IN REHEATER
      CPR=CPLF(PT,(TTO+TRO)/2.)
      TRO=TTO+(HRO-HGSPT)/CPR
      RQR=WRO/CRQR
      RLR=1./(1./RGR+1./RQT)
*
      B=TMA(RLR,WRO*CPR)
      TRM=B*TTO+(1.-B)*TRO
      QRR=RQR*(TTR-TRM)
      HRTV :WTI*HTI-WTO*HTO
      HRRV :WTO*HTO+QRR-WRO*HRO
      TM=TSAT(PM)
      HM=HPS(TM)
      QTR=RQT*(TM-TTR)
      WM=KVM*(PV-PM)*AMOV
      WMC=QTR/(HV-HM)
      WSL=SQRT((PE-PV)/KE)
      PV :(WSL-(WV+WM))/(VSL*RGV)
      PM :(WM-WMC)/(RGR*VMO)
      TTR :(QTR-QRR)/CTR
      WWW=WT-WTI
*
*     ------------------------------------------------------------
*
*     LOW-PRESSURE TURBINE
*
*     FUNCTIONS
```

```
*       SGS , SGSU , HGS , HFS
*       ( PSAT : PREMATURELY )
*
*     STATE VARIABLES
*       PL              STEAM PRESSURE IN INLET CHAMBER
*       HLP             ENTHALPY OF STEAM IN INLET CHAMBER
*
*     INPUT VARIABLES
*       PC              PRESSURE IN CONDENSER
*       WRO             STEAM FLOW INTO INLET CHAMBER
*       WS(1-3)         ----- LOAD BY STEAM COOLER 1,2 AND 3
*       TCON            TEMPERATURE OF STEAM IN CONDESER
*
*     PARAMETERS
*       KL              FLOW COEFFICIENT FOR TURBINE
*       VL              VOLUME OF TURBINE INLET CHAMBER
*       RGL             DIFFERENTIAL QUOTIENT OF RHO WITH RESP. TO PRESSUI
*       GL              EFFICIENCY OF TURBINE
*
*     ALGEBRAIC VARIABLES
*       WL              STEAM FLOW OUT OF INLET CHAMBER
*       WLO             ----- ---- --- -- TURBINE
*       SL              ENTROPY OF STEAM IN INLET CHAMBER    .
*       HC              SPECIFIC ENTHALPY OF STEAM OUT OF TURBINE
*
*     THE INPUT VARIABLE PC IS CALCULATED 'PREMATURELY'
      PC=PSAT(TCON)
*
      PR=PC/PL
      IF(PR.LT..7) THEN
        YL=1.
      ELSE
        YL=1.-(100./9.)*((PR-.7)**2.)
      END IF
      WL=KL*PL*YL
      PL :(WRO-WL)/(RGL*VL)
      HL=HLP/(VL*RGL*PL)
      HLP :WRO*HRO-WL*HL
      TL=TSAT(PL)
```

```
      SL=SGS(TL)+SGSU(PL,HL-HGS(TL))
      SFSPC=SFS(TCON)
      XC=(SL-SFSPC)/(SGS(TCON)-SFSPC)
      HC=XC*HGS(TCON)+(1.-XC)*HFS(TCON)
      HC=HL-GL*(HL-HC)
      WLO=WL-(WS(1)+WS(2)+WS(3))
*
*     ------------------------------------------------------------
*
*     CONDENSER
*
*     FUNCTIONS
*       HFS
*       ( PSAT : HAS BEEN USED PREMATURELY )
*
*     STATE VARIABLES
*       TCON          TEMPERATURE OF TUBE AND OF STEAM AND CONDENSATE
*       TCO           ---------- -- WATER AT OUTLET
*       MAC           MASS OF ACCUMULATED CONDENSATE
*
*     INPUT VARIABLES
*       WLO           STEAM FLOW FROM TURBINE
*       WCON          FLOW OF CONDENSATE OUT OF CONDENSER
*       WCW           ---- -- WATER IN TUBE
*       TCI           TEMPERATURE OF WATER AT INLET
*
*     PARAMETERS
*       BC            WEIGHT FACTOR AT CALCULATION OF MEAN TEMPERATURE
*       KCW           HEAT TRANSFER COEFFICIENT, FROM TUBES TO WATER
*       CPC           SPECIFIC HEAT CAPACITY OF WATER
*       CCW             TOTAL  ---- -------- -- -----
*       CCT             -----  ---- -------- -- TUBE
*
*     ALGEBRAIC VARIABLES
*       PC            STEAM PRESSURE IN CONDENSER
*       TCM           MEAN TEMPERATURE OF WATER
*
*     PC=PSAT(TCON)  HAS ALREADY BEEN CALCULATED
*
```

```
      TCM=BC*TCI+(1.-BC)*TCO
      QCT=WLO*(HC-HFS(TCON))
      QCW=KCW*(TCON-TCM)
      TCON :(QCT-QCW)/CCT
      TCO :(QCW+WCW*(TCI-TCO)*CPC)/CCW
      MAC :WLO-WCON
*
*-----------------------------------------------------------------------
*
*     ALGEBRAIC VARIABLES FOR THE TURBINE SECTION :
*           EXTRACTION PRESSURES AND ENTHALPIES
*           POWER PRODUCTION
*           WATER FLOWS FROM REHEATER
*
*     STATE VARIABLES
*       NONE
*
*     INPUT VARIABLES
*       PC          STEAM PRESSURE IN CONDENSER
*       PL          ----- -------- -- LP INLET CHAMBER
*       PT          ----- -------- -- REHEATER
*       PH          ----- -------- -- HP INLET CHAMBER
*       WH          ----- FLOW THROUGH HP TURBINE
*       WS(5)       ----- FLOWS, EXTRACTION
*       HC          SPEC. ENTHALPY OF STEAM OUT OF LP TURBINE
*       HL          ----- -------- -- ----- AT LP INLET CHAMBER
*       HH          ----- -------- -- ----- -- HP ----- -------
*       HHO         ----- -------- -- ----- OUT OF HP TURBINE
*       HSB(3)      ----- -------- -- -----, EXTRACTION
*       WT          FLOW OF  WET  STEAM TO REHEATER
*       WTI         ---- -- DRIED ----- -- MS
*       WMC         ---- -- CONDENSED STEAM
*       TT          TEMPERATURE OF STEAM IN MS
*       TM          ---------- -- CONDENSED STEAM
*
*     PARAMETERS
*       PFVR(3)
*       A(2)
*       HFVR(3)
```

```
*       HFVR5
*
*     ALGEBRAIC VARIABLES
*       PS(5)         STEAM PRESSURE
*       HSI(5)        SPEC. ENTHALPY OF STEAM
*       ET            POWER PRODUCTION, TOTAL
*       EH            ----- ----------, HP-TURBINE
*       EL            ----- ----------, LP-TURBINE
*       WRH(2)        FLOW  OF WATER FROM REHEATER
*       TRH(2)        TEMP. -- ----- ---- --------
* DO 200 J=1,3
        PS (J)=PC+PFVR(J)*(PL-PC)
        HSI(J)=HC+HFVR(J)*(HL-HC)
  200 CONTINUE
      PS (4)=PT
      HSI(4)=HHO
      PS (5)=PT+(A(1)+A(2)*WH)*(PH-PT)
      HSI(5)=HHO+HFVR5*(HH-HHO)
*
      EH=WH*(HH-HHO)-WS(5)*(HSI(5)-HHO)
      E=0.
      DO 300 J=1,3
        IF(WS(J).GT.0.) THEN
          H=HSI(J)
        ELSE
          H=HSB(J)
        END IF
        E=E+WS(J)*(H-HC)
  300 CONTINUE
      EL=WL*(HL-HC)-E
      ET=EH+EL
*
      WRH(1)=WT-WTI
      TRH(1)=TT
      WRH(2)=WMC
      TRH(2)=TM
*
      RETURN
*
```

```
*
      ENTRY TOUT
*
      WRITE(10,'(A)') '1'
      WRITE(10,1000) 'PE,PV,PM,PH,PT,PL,PC',PE,PV,PM,PH,PT,PL,PC
      WRITE(10,1000) 'HHP,HRTV,HRHV,HLP',HHP,HRTV,HRHV,HLP
      WRITE(10,1000) 'HV,HH,HHO,HM',HV,HH,HHO,HM
      WRITE(10,1000) 'HTI,HTO,HRO,HC',HTI,HTO,HRO,HC
      WRITE(10,1000) 'WRH(2),TRH(2)',WRH,TRH
      WRITE(10,1000) 'TE,TM,TT,TTR,TCON,TCO',TE,TM,TT,TTR,TCON,TCO
      WRITE(10,1000) 'TH,TRO,TRM,TCM',TH,TRO,TRM,TCM
      WRITE(10,1000) 'WV,WH,WT,WTI,WTO,WRO',WV,WH,WT,WTI,WTO,WRO
      WRITE(10,1000) 'WSL,WM,WMC,WWW,WL,WLO',WSL,WM,WMC,WWW,WL,WLO
      WRITE(10,1000) 'MAC,SH,SL,XHO',MAC,SH,SL,XHO
      WRITE(10,1000) 'KLR,KQR,CPR*1000,B',KLR,KQR,1000*CPR,B
      WRITE(10,1000) 'ET,EL,EH',ET,EL,EH
      WRITE(10,1000) 'EXTRACTION : P AND H',PS,HSI
      WRITE(10,*)
      WRITE(10,*)
      WRITE(10,*)
*
 1000 FORMAT(X,A22,10(F11.5))
*
      RETURN
      END
```

<u>Feedwater Preheater Chain. Listing</u>

```
& *
&
& %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
& %%%%%%%%%%%%%%%%%%%%%%%         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
& %%%%%%%%%%%%%%                            %%%%%%%%%%%%%%%%%%%%%
& %%%%%%%%%%           PREHEATER CHAIN          %%%%%%%%%%%%%%%%%
& %%%%%%%%%%%%%%                            %%%%%%%%%%%%%%%%%%%%%
& %%%%%%%%%%%%%%%%%%%%%%%         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
& %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
&
```

```
& *
!     MODUL          CH
&     =====
&
&
!STV       PFV4,PFV5,WS(3),H(5),TD(5),TTS(5),TIS(5),TF(5),TOS(5),
!          Z(5),
!          TM(3),
!          X(5),
!          TTH(4),TW(4),TIH(4),TC(4),TOH(4);
!INP       PS(5),HSI(5),
!          WTOTIN,
!          TFI,
!          WRH(2),TRH(2);
!PAR       HSET(5);
!TXT       REPORT;
!ALG       WFI,
!          TFM(5),WDO(5),HSE(3),
!          Y(5),
!          WM(3),
!          TOD(5),
!          TWM(4),TCM(4),
!          WFV4,WFV5;
&
      REAL           WV(5),W51,W52,DUMMY
      REAL           CV(5),RHOF(5),SPLIT
C *
C *
C *   ------------------------------------------------------------
C *
C *   FLOW COEFFICIENTS
      DATA CV /630.,630.,800.,400.,210./
C *   FEED WATER DENSITY, IN DELAY TUBES
      DATA RHOF /982.6,965.8,952.2,920.2,888.7/
C *   RATIO OF WATER FROM SCB.4 FLOWING THROUGH HX.4
      DATA SPLIT /.145/
C *
C *   ------------------------------------------------------------
C *
```

```
*     FLOW INTO THE CHAIN (WFI) IS DETERMINED BY THE FLOW OF DRAINAGE
*     OUT OF THE FIRST STEAM COOLER (WDO(1)) AND THE FLOW OF WATER
*     EXTRACTED FROM THE CHAIN (WTOTIN=W_TOT_OUT)
      WFI=WTOTIN-WDO(1)
*
*
*     ******************        STAGE 1          **********************
*
*     OUTPUT VALVE CONTROLLED BY PI-REGULATOR
      WDO(1)=CV(1)*Y(1)
*
*
* >>> STEAM COOLER NO 1
*
!     MACRO          SCA
&     =====
&
!STV       WS   : WS(1)     , H     : H(1)      ,
!          TD     : TD(1)     , TT     : TTS(1)    ,
!          TI     : TIS(1)    , TF     : TF(1)     , TO     : TOS(1)    ;
!INP       PS     : PS(1)     , HSI    : HSI(1)    ,
!          TDI    : TW(1)     , WDI    : WDO(2)    , WDO    : WDO(1)    ,
!          WF     : WFI       , TFI    : TFI       ;
!PAR       TAU    = .25       , KST    = 11.55     , KTF    = 5.775     ,
!          CT     = 3.02      ,
!          CI     = 1         , CF     = 10        , CO     = 1         ,
!          CPF    = 4.2E-3    , CPD    = .418E-2   ,
!          RHOD   = 982       , AD     = 6.28      ;
!ALG       HSB    : HSB(1)    , TFM    : TFM(1)    ;
!     END
&
*
* >>> CONTROL OF WATER LEVEL
*
*
!     MACRO          PI
&     =====
&
!STV       Z:Z(1);
```

```
!INP        X:H(1),XS:HSST(1);
!PAR        X0=.9,XM=2.6,PB=0.5,TAU=360.,G=1.25;
!ALG        Y:Y(1);
!TXT        REPORT:REPORT;
!      END
*
* >>> TUBES TO DRAIN COOLER
*
*
!      MACRO          MIX
&      =====
&
!STV        T:TM(1);
!INP        W1:WDO(1),T1:TD(1),W2:WFI,T2:TOS(1),RHO:RHOF(1);
!PAR        V=3;
!ALG        WO:WM(1);
!      END
C *
*      VOLUME FLOW THROUGH "DELAY TUBE"
       WV(1)=WM(1)/RHOF(1)
*
!      MACRO          TIDEL
&      =====
&
!STV        X:X(1);
!INP        T:TIME,V:WV(1),TI:TM(1);
!PAR        L=26.8;
!ALG        TO:TOD(1);
!      END
C *
*      ******************        STAGE 2          ************************
*
* >>> DRAIN COOLER NO 1
*
!      MACRO          HX
&      =====
&
!STV        TT:TTH(1),TW:TW(1),TI:TIH(1),TC:TC(1),TO:TOH(1);
!INP        WW:WDO(2),WC:WM(1),TWI:TD(2),TCI:TOD(1);
```

```
!PAR        KWT=3.74,KTC=3.74,CT=2.8,CW=84,CCI=0.922,CC=9.22,CCO=.922,
!           CPW=4.19E-3,CPC=42E-4;
!ALG        TWM   : TWM(1)    , TCM   : TCM(1)    ;
!      END
&
*
* >>> STEAM COOLER NO 2
*
*      OUTPUT VALVE CONTROLLED BY PI-REGULATOR
       WDO(2)=CV(2)*Y(2)
*
!      MACRO          SCA
&      =====
&
!STV        WS:WS(2),H:H(2),TD:TD(2),TT:TTS(2),
!           TI:TIS(2),TF:TF(2),TO:TOS(2);
!INP        PS:PS(2),HSI:HSI(2),TDI:TW(2),WDI:WDO(3),WDO:WDO(2),
!           WF:WM(1),TFI:TOH(1);
!PAR        TAU=.25,KST=15,KTF=7.5,CT=6.12,CI=2,CF=20,CO=2,
!           CPF=4.2E-3,CPD=.419E-2,RHOD=962,AD=4.54;
!ALG        HSB:HSB(2),TFM:TFM(2);
!      END
&
*
* >>> CONTROL OF WATER LEVEL
*
*
!      MACRO          PI
&      =====
!STV        Z:Z(2);
!INP        X:H(2),XS:HSET(2);
!PAR        X0=.0,XM=1.4,PB=0.7,TAU=38.4,G=2.50;
!ALG        Y:Y(2);
!TXT        REPORT:REPORT;
!      END
*
* >>> TUBES TO DRAIN COOLER
*
C *
```

```
*     VOLUME FLOW THROUGH "DELAY TUBE"
      WV(2)=WM(1)/RHOF(2)
*
!     MACRO          TIDEL
&     =====
!STV       X:X(2);
!INP       T:TIME,V:WV(2),TI:TOS(2);
!PAR       L=9.84;
!ALG       TO:TOD(2);
!     END
*
*     ******************       STAGE 3          **********************
*
* >>> DRAIN COOLER NO 2
&
!     MACRO          HX
&     =====
!STV       TT:TTH(2),TW:TW(2),TI:TIH(2),TC:TC(2),TO:TOH(2);
!INP       WW:WDO(3),WC:WM(1),TWI:TD(3),TCI:TOD(2);
!PAR       KWT=4.24,KTC=4.24,CT=1.56,CW=48,CCI=0.506,CC=5.06,CCO=.506,
!          CPW=4.21E-3,CPC=42E-4;
!ALG       TWM   : TWM(2)    , TCM   : TCM(2)    ;
!     END
&
*
* >>> STEAM COOLER NO 3
C *
*     OUTPUT VALVE CONTROLLED BY PI-REGULATOR
      WDO(3)=CV(3)*Y(3)
*
!     MACRO          SCA
&     =====
!STV       WS:WS(3),H:H(3),TD:TD(3),TT:TTS(3),
!          TI:TIS(3),TF:TF(3),TO:TOS(3);
!INP       PS:PS(3),HSI:HSI(3),TDI:TW(3),WDI:WDO(4),WDO:WDO(3),
!          WF:WM(1),TFI:TOH(2);
!PAR       TAU=.25,KST=15.45,KTF=7.725,CT=7.26,CI=2.34,CF=23.4,CO=2.34,
!          CPF=4.2E-3,CPD=.421E-2,RHOD=949,AD=9.04;
!ALG       HSB   : HSB(3)    , TFM   : TFM(3)    ;
```

```
  !     END
  &
*
* >>> CONTROL OF WATER LEVEL
*
*
!     MACRO          PI
&     =====
!STV        Z:Z(3);
!INP        X:H(3),XS:HSET(3);
!PAR        X0=.0,XM=1.35,PB=0.5,TAU=90.0,G=1.25;
!ALG        Y:Y(3);
!TXT        REPORT:REPORT;
!     END
*
&
*
* >>> TUBES TO DRAIN COOLER
*
C *
*     VOLUME FLOW THROUGH "DELAY TUBE"
      WV(3)=WM(1)/RHOF(3)
*
!     MACRO          TIDEL
&     =====
!STV        X:X(3);
!INP        T:TIME,V:WV(3),TI:TOS(3);
!PAR        L=16.74;
!ALG        TO:TOD(3);
!     END
*
*     *****************       STAGE 4          ************************
*
* >>> DRAIN COOLER NO 3
*
&
!     MACRO          HX
&     =====
!STV        TT:TTH(3),TW:TW(3),TI:TIH(3),TC:TC(3),TO:TOH(3);
```

```
!INP        WW:WDO(4),WC:WM(1),TWI:TD(4),TCI:TOD(3);
!PAR        KWT=4.14,KTC=4.14,CT=3.14,CW=52,CCI=0.800,CC=8.00,CCO=.800,
!           CPW=4.25E-3,CPC=42E-4;
!ALG        TWM   : TWM(3)    , TCM   : TCM(3)    ;
!      END
&
*      MIX SOME WATER FROM PREHEATER INTO DRAIN FROM DRAIN COOLER 4
*      THIS IS PART OF STAGE 5 BUT BECAUSE OF ORDER OF CALCULATION
*      IT MUST BE DONE HERE.
C *
*
!      MACRO          MIX
&      =====
!STV        T:TM(2);
!INP        W1:WDO(5),T1:TW(4),W2:WRH(1),T2:TRH(1),RHO:RHOF(4);
!PAR        V=3;
!ALG        WO:WM(2);
!      END
*
*
* >>> STEAM COOLER NO 4
*
*      OUTPUT VALVE CONTROLLED BY PI-REGULATOR
       WDO(4)=CV(4)*Y(4)
*
*      VALVE CALCULATION
       WFV4=AMAX1(400.*(PS(4)-PFV4),0.)
*
!      MACRO          SCB
&      =====
&
!STV        PS:PFV4,H:H(4),TD:TD(4),TT:TTS(4),
!           TI:TIS(4),TF:TF(4),TO:TOS(4);
!INP        WS:WFV4,HSI:HSI(4),TDI:TM(2),WDI:WM(2),WDO:WDO(4),
!           WF:WM(1),TFI:TOH(3);
!PAR        KST=19.56,KTF=9.780,CT=8.47,CI=2.14,CF=21.4,CO=2.14,
!           CPF=4.22E-3,CPD=.425E-2,RHOD=916,AD=4.02,
!           RG=0.47,V=80;
!ALG        HSB   : DUMMY     , TFM   : TFM(4)    ;
```

```
!     END
*
* >>> CONTROL OF WATER LEVEL
*
!     MACRO          PI
&     =====
!STV       Z:Z(4);
!INP       X:H(4),XS:HSET(4);
!PAR       X0=.394,XM=1.194,PB=1.0,TAU=180.,G=2.50;
!ALG       Y:Y(4);
!TXT       REPORT:REPORT;
!     END
*
* >>> TUBES TO DRAIN COOLER
*
C *
*     VOLUME FLOW THROUGH "DELAY TUBE"
      WV(4)=WM(1)/RHOF(4)
*
!     MACRO          TIDEL
&     =====
!STV       X:X(4);
!INP       T:TIME,V:WV(4),TI:TOS(4);
!PAR       L=3.66;
!ALG       TO:TOD(4);
!     END
*
*     ******************       STAGE 5          **********************
*
*     FLOW IS SPLIT INTO TWO FLOWS : ONE GOES DIRECTLY TO STEAM
*     COOLER AND ONE TO DRAIN COOLER
*
      W51=SPLIT*WM(1)
      W52=(1-SPLIT)*WM(1)
*
* >>> DRAIN COOLER NO 4
*
!     MACRO          HX
&     =====
```

```
!STV      TT:TTH(4),TW:TW(4),TI:TIH(4),TC:TC(4),TO:TOH(4);
!INP      WW:WDO(5),WC:W51,TWI:TD(5),TCI:TOD(4);
!PAR      KWT=2.08,KTC=2.08,CT=1.29,CW=28,CCI=0.312,CC=3.12,CCO=.312,
!         CPW=4.34E-3,CPC=42.2E-4;
!ALG      TWM   : TWM(4)    , TCM   : TCM(4)    ;
!     END
*
*     REMIX THE TWO FLOWS
*
!     MACRO          MIX
&     =====
!STV      T:TM(3);
!INP      W1:W51,T1:TOH(4),W2:W52,T2:TOD(4),RHO:RHOF(4);
!PAR      V=3;
!ALG      WO:WM(3);
!     END
*
* >>> STEAM COOLER NO 5
C *
*     OUTPUT VALVE CONTROLLED BY PI-REGULATOR
      WDO(5)=CV(5)*Y(5)
*
*     VALVE CALCULATION
      WFV5=AMAX1(400.*(PS(5)-PFV5),0.)
*
!     MACRO          SCB
&     =====
!STV      PS:PFV5,H:H(5),TD:TD(5),TT:TTS(5),
!         TI:TIS(5),TF:TF(5),TO:TOS(5);
!INP      WS:WFV5,HSI:HSI(5),TDI:TRH(2),WDI:WRH(2),WDO:WDO(5),
!         WF:WM(3),TFI:TM(3);
!PAR      KST=18.66,KTF=9.33 ,CT=8.47,CI=2.05,CF=20.5,CO=2.05,
!         CPF=4.28E-3,CPD=.434E-2,RHOD=882,AD=4.02,
!         RG=0.47,V=80;
!ALG      HSB:DUMMY , TFM   : TFM(5)    ;
!     END
*
* >>> CONTROL OF WATER LEVEL
*
```

```
!     MACRO          PI
&     =====
!STV       Z:Z(5);
!INP       X:H(5),XS:HSET(5);
!PAR       X0=.394,XM=1.194,PB=1.0,TAU=19.2,G=1.25 ;
!ALG       Y:Y(5);
!TXT       REPORT:REPORT;
!     END
C *
* >>> TUBES OUT OF CHAIN
*
*     VOLUME FLOW THROUGH "DELAY TUBE"
      WV(5)=WM(3)/RHOF(5)
*
!     MACRO          TIDEL
&     =====
!STV       X:X(5);
!INP       T:TIME,V:WV(5),TI:TOS(5);
!PAR       L=15.6;
!ALG       TO:TOD(5);
!     END
&
      RETURN
*
      ENTRY COUT
*
*     PROFILE
      WRITE(10,1000) 'WFI,WM(1-3),WRH(1-2)',WFI,WM,WRH
      WRITE(10,1000) 'TFI,TM(1-3),TRH(1-2)',TFI,TM,TRH
      WRITE(10,1000) 'H',H
      WRITE(10,1000) 'STEAM FLOW',WS,WFV4,WFV5
      WRITE(10,1000) 'STEAM PRESSURE',PS(1),PS(2),PS(3),PFV4,PFV5
      WRITE(10,1000) 'STEAM TEMP.',TSAT(PS(1)),TSAT(PS(2)),
     &      TSAT(PS(3)),TSAT(PFV4),TSAT(PFV5)
      WRITE(10,1000) 'TTS',TTS
      WRITE(10,1000) 'TIS',TIS
      WRITE(10,1000) 'TFM',TFM
      WRITE(10,1000) 'TF',TF
      WRITE(10,1000) 'TOS',TOS
```

```
      WRITE(10,1000) 'TD',TD
      WRITE(10,1000) 'TWM',TWM
      WRITE(10,1000) 'TW',TW
      WRITE(10,1000) 'TIH',TIH
      WRITE(10,1000) 'TCM',TCM
      WRITE(10,1000) 'TC',TC
      WRITE(10,1000) 'TOH',TOH
      WRITE(10,1000) 'TTH',TTH
      WRITE(10,1000) 'WDO',WDO
      WRITE(10,1000) 'TOD',TOD
      WRITE(10,1000) 'WFV4,WFV5,PFV4,PFV5',WFV4,WFV5,PFV4,PFV5
      WRITE(10,*) ' '
      RETURN
*
 1000 FORMAT(X,A22,10(F11.5))
*
      END
```

Connecting System, Listing

```
*
!SYS TST, 3
!SYS TU,CH,
!    CON;
! DCL
      REAL TAU
      LOGICAL AABNET
      DATA TAU /10./
! END
*
*
! MOD TU
      PE=PE.CON
      WS1=WS1.CH
      WS2=WS2.CH
      WS3=WS3.CH
      WS4=WFV4.CH
      WS5=WFV5.CH
```

```
      XV=XV.CON+AMIN1(1.,T/RISET1.CON)*(XVF.CON-XV.CON)
      ALTV=ALTV.CON
      AMOV=AMOV.CON
      WCON=WFI.CH
      WCW=WCW.CON+AMIN1(1.,T/RISET2.CON)*(WCWF.CON-WCW.CON)
      TCI=TCI.CON
      HSB1=HSB1.CH
      HSB2=HSB2.CH
      HSB3=HSB3.CH
! END
*
!MOD CH
      PS1=PS1.TU
      PS2=PS2.TU
      PS3=PS3.TU
      PS4=PS4.TU
      PS5=PS5.TU
      HSI1=HSI1.TU
      HSI2=HSI2.TU
      HSI3=HSI3.TU
      HSI4=HSI4.TU
      HSI5=HSI5.TU
      WTOTIN=WTOTIN.CON
      TFI=TCON.TU
      WRH1=WRH1.TU
      WRH2=WRH2.TU
      TRH1=TRH1.TU
      TRH2=TRH2.TU
!END
      IF(REPORT.NE.' ') THEN
        INQUIRE(98,OPENED=AABNET)
        IF(.NOT. AABNET) OPEN(98,STATUS='SCRATCH')
        WRITE(98,'(A)') REPORT
        WRITE(98,6000) T,NR
 6000   FORMAT(5X,'T=',F9.5,'  NR=',I10)
        REPORT=' '
      END IF
*
      WTOTIN : (WSL.TU-WTOTIN.CON)/TAU
```

```
!OUT
      CALL TOUT
      CALL COUT
      WRITE(10,9000) 'XV,WTOTIN ',XV.CON,WTOTIN.CON
      INQUIRE(98,OPENED=AABNET)
      IF(AABNET) THEN
        ENDFILE(98)
        REWIND(98)
 7000   CONTINUE
          READ(98,'(A)',END=8000) REPORT
          WRITE(10,'(X,A)') REPORT
        GOTO 7000
 8000   CONTINUE
        CLOSE(98)
      END IF
*
 9000 FORMAT(X,A22,10F11.5)
*
```

APPENDIX D

LISTING OF THE LIST FILE THAT WAS USED FOR
SIMULATION OF A PART OF A POWER PLANT

List file

MODULE: TU TURBINES, SUPERHEATER AND CONDENSER

STATE VARIABLES 13

NAME	STU	DIMENSION	TEXT
PV	1	BAR	PRESSURE BEFORE VALVE
PH	2	BAR	-------- IN HP-TURBINE INLET CHAMBER
HHP	3	MJ	ENTHALPY OF WET STEAM IN --- -------
PT	4	BAR	PRESSURE IN LP-TURBINE INLET CHAMBER
HRTV	5	MJ	ENTHALPY OF STEAM BEFORE SUPER HEATER
HRHV	6	MJ	-------- -- ----- IN ----- ------
PM	7	BAR	STEAM PRESSURE IN TUBES
TTR	8	C	TEMPERATURE OF TUBES IN REHEATER
PL	9	BAR	PRESSURE IN LP-TURBINE INLET CHAMBER
HLP	10	MJ	ENTHALPY OF STEAM IN INLET CHAMBER
TCON	11	C	TEMPERATURE IN CONDENSER
TCO	12	C	---------- -- WATER AT OUTLET (COOLANT)
MAC	13	KG	MASS OF ACCUMULATED CONDENSATE

ALGEBRAIC VARIABLES 41

NAME	ATU	DIMENSION	TEXT
WSL	1	KG/S	STEAM FLOW IN STEAM LINE
WV	2	KG/S	----- FLOW THROUGH HP-VALVE
WH	3	KG/S	----- ---- ------- HP-TURBINE
WT	4	KG/S	----- ---- TO REHEATER
SH	5	MJ	ENTROPY OF STEAM IN INLET CHAMBER (HP)

HV	6	MJ/KG	SPECIFIC ENTHALPY OF STEAM THROUGH VALVE
HH	7	MJ/KG	-------- -------- -- ----- IN INLET CHMB
XHO	8		STEAM QUALITY
WTI	9	KG/S	FLOW OF DRIED STEAM INTO MOISTURE SEP.
WTO	10	KG/S	---- -- ----- ----- OUT OF ------ ----
WRO	11	KG/S	---- -- SUPER HEATED STEAM OUT OF REHEAT
HTI	12	MJ/KG	SPEC. ENTHALPY OF DRIED STEAM INTO MS
HTO	13	MJ/KG	----- -------- -- ----- ----- OUT OF MS
HRO	14	MJ/KG	----- -------- -- SUPER HEATED STEAM,OUT
TRO	15	C	----- -- ----- OUT OF REHEATER
TRM	16	C	MEAN TEMPERATURE OF STEAM IN REHEATER
WM	17	KG/S	FLOW OF CONDENSING STEAM THROUGH VALVE
WWW	18	KG/S	---- -- WATER REMOVED FROM WET STEAM
WL	19	KG/S	STEAM FLOW OUT OF INLET CHAMBER
WLO	20	KG/S	----- ---- --- -- TURBINE
SL	21	MJ	ENTROPY OF STEAM IN INLET CHAMBER
HC	22	MJ/KG	SPEC. ENTHALPY OF STEAM OUT OF LP-TURBIN
PC	23	BAR	STEAM PRESSURE IN CONDENSER
TCM	24	C	MEAN TEMPERATURE OF WATER (COOLANT)
PS(5)	25-29	BAR	PRESSURE OF STEAM AT LOAD TUBES
HSI(5)	30-34	MJ/KG	SPEC. ENTHALPY OF STEAM AT LOAD TUBES
ET	35	MJ/S	POWER PRODUCTION, TOTAL
EH	36	MJ/S	----- ----------, HP-TURBINE
EL	37	MJ/S	----- ----------, LP-TURBINE
WRH(2)	38-39	KG/S	FLOW OF WATER FROM REHEATER
TRH(2)	40-41	C	TEMPERATURE OF WATER FROM REHEATER

INPUT VARIABLES 15

NAME	XIU	DIMENSION	TEXT
PE	1	BAR	STEAM PRESSURE BEFORE STEAM LINE
WS(5)	2-6	KG/S	----- LOAD
XV	7		VALVE OPENING, HP-VALVE
ALTV	8		----- -------, REHEATER TO LP-TURBINE
AMOV	9		----- -------, STEAM LINE TO REHEATER
WCON	10	KG/S	FLOW LOAD OF CONDENSATE

WCW	11	KG/S	FLOW OF WATER IN TUBE (COOLANT)
TCI	12	C	TEMPERATURE OF WATER AT INLET (COOLANT)
HSB(3)	13-15	MJ/KG	SPEC. ENTHALPY OF STEAM IF BACKWARD FLOW

PARAMETERS 0

MODULE: CH PRE-HEATER CHAIN

==========

STATE VARIABLES 68

NAME	SCH	DIMENSION	TEXT
PFV4	1	BAR	STEAM PRESSURE IN STEAM COOLER 4
PFV5	2	BAR	----- -------- -- ----- ------ 5
WS(3)	3-5	KG/S	----- LOAD
H(5)	6-10	M	HEIGHT OF DRAINAGE IN TANKS
TD(5)	11-15	C	TEMPERATURE OF DRAINAGE
TTS(5)	16-20	C	---------- -- TUBES IN STEAM COOLERS
TIS(5)	21-25	C	---------- -- WATER IN INLET CHAMBER
TF(5)	26-30	C	---------- -- ----- OUT OF TUBES
TOS(5)	31-35	C	---------- -- ----- IN OUTLET CHAMBER
Z(5)	36-40		ERROR INTEGRAL, PI-REGULATOR
TM(3)	41-43	C	TEMPERATURE OF WATER OUT OF MIXERS
X(5)	44-48	M**3	TOTAL VOLUME THROUGH DELAY TUBES
TTH(4)	49-52	C	TEMPERATURE OF TUBES IN HEAT EXCHANGERS
TW(4)	53-56	C	---------- -- WARM WATER IN HX
TIH(4)	57-60	C	---------- -- WATER IN INLET CHAMBER
TC(4)	61-64	C	---------- -- COLD WATER, OUT OF TUBES
TOH(4)	65-68	C	---------- -- WATER IN OUTLET CHAMBER

ALGEBRAIC VARIABLES 37

NAME	ACH	DIMENSION	TEXT
WFI	1	KG/S	FEED WATER LOAD, FROM CONDENSER
TFM(5)	2-6	C	MEAN TEMPERATURE OF FEED WATER, ST-COOL
WDO(5)	7-11	KG/S	FLOW OF DRAINAGE OUT OF TANKS

HSB(3)	12-14	MJ/KG	SPECIFIC ENTHALPY OF STEAM, BACKW. FLOW
Y(5)	15-19		OUTPUT FROM PI-REGULATORS
WM(3)	20-22	KG/S	FLOW OUT OF MIXERS
TOD(5)	23-27	C	TEMPERATURE OF WATER LEAVING MIXERS
TWM(4)	28-31	C	MEAN TEMPERATURE OF WARM WATER, HX
TCM(4)	32-35	C	---- ---------- -- COLD -----, --
WFV4	36	KG/S	STEAM FLOW THROUGH VALVE
WFV5	37	KG/S	STEAM FLOW THROUGH VALVE

INPUT VARIABLES 16

NAME	XCH	DIMENSION	TEXT
PS(5)	1-5	BAR	STEAM PRESSURE IN STEAM COOLERS
HSI(5)	6-10	MJ/KG	SPECIFIC ENTHALPY OF STEAM FROM TURBINES
WTOTIN	11	KG/S	FEED WATER FLOW TO REACTOR
TFI	12	C	TEMPERATURE OF WATER FROM CONDENSER
WRH(2)	13-14	KG/S	FLOW OF WATER TO DRAINAGE TANKS 4 AND 5
TRH(2)	15-16	C	TEMPERATURE OF WATER TO DRAINAGE TANKS

PARAMETERS 5

NAME	XCH	DIMENSION	TEXT
HSET(5)	1-5	M	SET POINT TO PI-REGULATORS, TANK-LEVELS

MODULE: CON TURBINE SECTION CONNECTED TO PREHEATER CHAIN

STATE VARIABLES 1

NAME	SCON	DIMENSION	TEXT
WTOTIN	1	KG/S	FEED WATER FLOW TO REACTOR

ALGEBRAIC VARIABLES 0

INPUT VARIABLES 0

PARAMETERS 10

NAME	PCON	DIMENSION	TEXT
PE	1	BAR	STEAM PRESSURE IN REACTOR
XV	2		VALVE OPENING, HP-VALVE
ALTV	3		----- -------, REHEATER TO LP-TURBINE
AMOV	4		----- -------, STEAM LINE TO REHEATER
WCW	5	KG/S	FLOW OF COOLANT IN CONDENSER
TCI	6	C	TEMP.-- ------- -- ---------
XVF	7		XV, FINAL VALUE AFTER RISE TIME ELAPSE
WCWF	8		WCW, FINAL VALUE AFTER RISE TIME ELAPSE
RISET(2)	9-10	S	RISE TIMES, FOR XV AND WCW

APPENDIX E

LISTING OF THE INPUT FILE THAT WAS USED FOR SIMULATION OF A PART OF A POWER PLANT

Input File

```
&            TURBINE SECTION CONNECTED TO PREHEATER CHAIN
&
&
*SYST
TST          3
TU           CH          CON
*INPT
  TRANS/3            FLOW OF SEAWATER DECREASED 50%
*INCO
TU           13          41
 6.76920E+01 5.86109E+01 8.26789E+02 5.08102E+00 1.85655E+03 1.67998E+03
 6.72018E+01 2.58673E+02 4.72896E+00 9.19015E+02 3.64166E+01 1.97921E+01
 0.0
 8.35460E+02 7.61941E+02 7.61941E+02 6.70375E+02 5.86353E+00 2.76596E+00
 2.76596E+00 8.51744E-01 5.73859E+02 5.73859E+02 5.73859E+02 2.73741E+00
 2.73741E+00 2.93119E+00 2.36184E+02 2.01704E+02 7.35192E+01 9.65183E+01
 5.73859E+02 5.08372E+02 7.23665E+00 2.31516E+00 6.06714E-02 1.44267E+01
 2.37600E-01 8.58949E-01 1.55452E+00 5.08102E+00 1.11923E+01 2.49935E+00
 2.66198E+00 2.74083E+00 2.43574E+00 2.53249E+00 0.0         0.0
 0.0         9.65183E+01 7.35192E+01 1.52288E+02 2.83166E+02
CH           68          37
 4.95237E+00 1.10921E+01 1.74899E+01 3.00759E+01 1.79212E+01 1.75000E+00
 4.20000E-01 4.05000E-01 7.94000E-01 7.94000E-01 6.36974E+01 9.42445E+01
 1.12385E+02 1.51627E+02 1.85187E+02 5.93052E+01 9.05211E+01 1.09108E+02
 1.46450E+02 1.78930E+02 3.64166E+01 6.95126E+01 9.50283E+01 1.19036E+02
 1.50010E+02 5.90242E+01 8.89151E+01 1.08158E+02 1.46274E+02 1.78325E+02
 5.90242E+01 8.89151E+01 1.08158E+02 1.46274E+02 1.78325E+02 4.15348E-01
 1.96570E-01 2.79522E-01 2.61601E-01 4.32837E-01 6.08538E+01 1.55632E+02
 1.50010E+02 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
```

```
 7.26602E+01 9.64349E+01 1.21694E+02 1.65330E+02 7.08227E+01 9.41567E+01
 1.17293E+02 1.58473E+02 6.08538E+01 8.89151E+01 1.08158E+02 1.46274E+02
 6.95126E+01 9.50282E+01 1.19037E+02 1.72042E+02 6.95126E+01 9.50283E+01
 1.19036E+02 1.72042E+02
 5.08372E+02 5.09466E+01 8.14434E+01 1.03144E+02 1.36631E+02 1.68079E+02
 3.27088E+02 3.09597E+02 2.79522E+02 2.61601E+02 1.13620E+02 0.00000E+00
 0.00000E+00 0.00000E+00 5.19187E-01 4.91425E-01 3.49404E-01 6.54002E-01
 5.41047E-01 8.35460E+02 2.10138E+02 8.35460E+02 6.08538E+01 8.89151E+01
 1.08158E+02 1.46274E+02 1.78325E+02 8.07841E+01 1.01494E+02 1.30915E+02
 1.71664E+02 6.45364E+01 9.13758E+01 1.12474E+02 1.58997E+02 5.14628E+01
 4.01006E+01
CON            1            0
 8.35460E+02
*STIC
*DATA
HSET1.CH    = 1.75000    HSET2.CH    = .420000    HSET3.CH    = .405000
HSET4.CH    = .794000    HSET5.CH    = .794000
XV.CON      = 0.49328    XVF.CON     = 0.49328    RISET1.CON = 10.
PE.CON      = 70.0000    ALTV.CON    = 1.00000    AMCV.CON    = 1.00000
WCW.CON     = 22200.0    WCWF.CON    = 11100.0    RISET2.CON = 5.
TCI.CON     = 8.00000
DATA.END
*CHCK
0.001        0.050        0.06000     0.001000       200         HEUN
&                         DT=1/16 WAS OBSERVED AS MAXIMUM STEP LENGTH
*TIME
 298
*REFV
 MAC.TU      MAC.TU       1000.
REFV.END
*PRNT
 .500000      20.0000      1.00000      50.0000      2.00000
WSL.TU       WTOTIN.CON   ET.TU        TOD5.CH
WS1.CH       WS2.CH       WS3.CH       WFV4.CH      WFV5.CH
TOS1.CH      TOS2.CH      TOS3.CH      TOS4.CH      TOS5.CH
PRNT.END
*PLOT
*DELY
*INPD
```

```
&STST
&1000
&JACO
&10
*ENDE
```

APPENDIX F

DESCRIPTION OF FUNCTIONS AND SUBMODULES USED FOR SIMULATION OF A SUGAR FACTORY

In this appendix the functions and the submodules that are part of the library which was used in the simulation of a part of a sugar factory are described. The functions in this appendix include a library of water functions, i.e. functions describing the thermodynamic properties of water and steam. When this library is used it will be indicated in the prerequisites by writing the name of the library: H2OFUNCTION. These functions will not be described here. There are a few functions that are being used also by the library with submodules used for simulation of power plants which will be described here. Finally the functions to be described here include some sugar functions, i.e. functions describing the thermodynamic properties of sugar. The description of the functions will follow the description of the submodules.

1. FLASH DRUM

1.1. Name

AFSP

1.2. Description

The flash drum is a tank with condensate which is being superheated by an inlet flow of condensate from steam at a higher pressure than in the flash drum. The superheating results in

evaporation and this steam is leaving the flash drum, i.e. a dynamic calculation of the pressure is not made . The level of condensate is controlled ideally so the volume of condensate is constant.

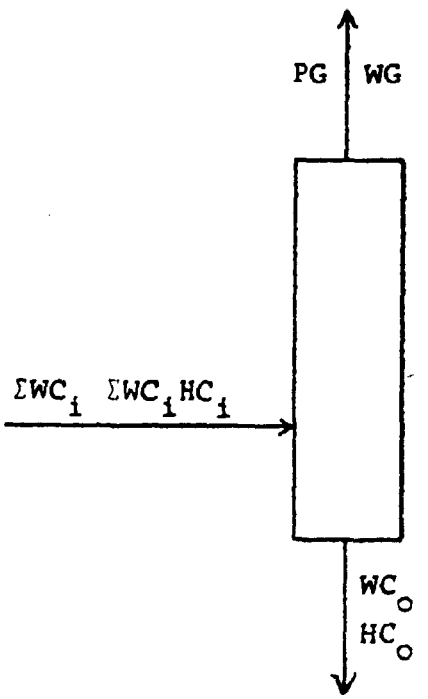
1.3. Characteristics

The evaporation rate is calculated using an empirical relation including the degree of superheating and the mass of condensate. If there are more than one inlet flow of condensate the total flow and the total enthalpy must be calculated in advance.

1.4. Prerequisites

The library H2OFUNCTIONS must be available.

1.5. Symbols



1.6. Identifiers

Name	Type	Dimension	Description
SWC_i	INP	kg/s	ΣWCi, sum of inlet flows
$SWCHC_i$	INP	MW	ΣWCiHCi, sum of inlet enthalpies
PG	INP	bar	Steam pressure
WG	ALG	kg/s	Evaporation rate
WC_o	ALG	kg/s	Outlet flow of condensate
HC_o	STV	MJ/kg	Specific enthalpy of condensate
V	PAR	m3	Volume of condensate

1.7. Model

The steam is saturated at pressure PG, so

```
TG=TSAF(PG)
HG=HGSF(TG)
```

are the temperature and the specific enthalpy of the steam. By assuming that the specific heat capacity of the condensate does not depend on the temperature the following expression is found for the temperature of the condensate

$$TC_o = \frac{HC_o}{c_p}$$

where

$$c_p = CPFSF(TG)/1000.$$

is the specific heat capacity of the condensate. The following empirical expression is used for calculation of the evaporation rate

$$WG = 0.008\ M\ (TC_o - TG)^2$$

where

$M = V \cdot RFSF(TG)$

RFSF is the density of the condensate.

The evaporation rate is zero if the condensate is not superheated, i.e. if $(TC_o - TG) < 0$.

Since the level of condensate is ideally controlled the outlet flow of condensate is

$$WC_o = SWC_i - WG$$

Applying conservation of energy on the condensate the following expression is found

$$\dot{HC}_o = \frac{SWCHC_i - WC_o \cdot HC_o - WG \cdot HG}{M}$$

2. PREHEATER

2.1. Name

VV

2.2. Description

A flow of water with sugar, sugar liquor, is heated by condensing steam. The liquor is flowing inside tubes and the steam is condensing outside. The condensate is flowing out of the preheater.

2.3. Characteristics

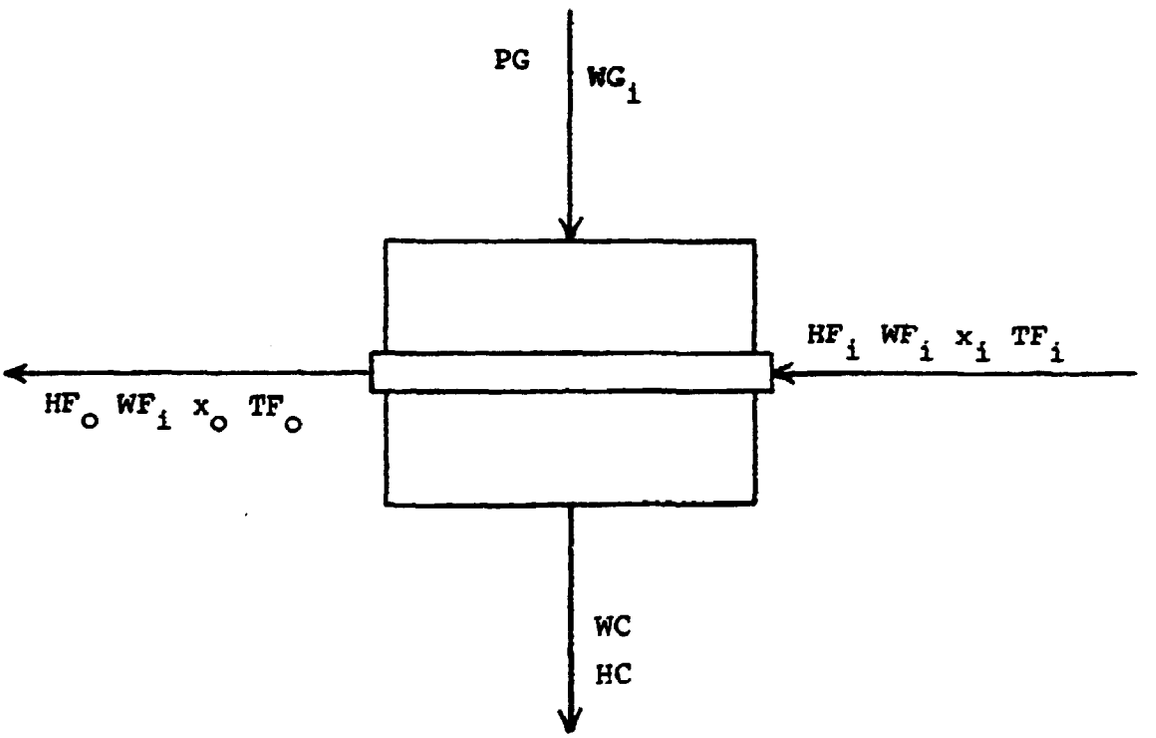
The pressure dynamics is ignored in this model. The sugar liquor

is lumped to one compartment the temperature of which is being calculated as a weighted sum of inlet and outlet temperature.

2.4. Prerequisites

The library H2OFUNCTIONS must be available. The sugar functions TSAFTH and CPSAFT must be available and the function TMA for calculation of a weight factor must be available.

2.5. Symbols



2.6. Identifiers

Name	Type	Dimension	Description
WF_i	INP	kg/s	Flow of sugar liquor, in
HF_i	INP	MJ/kg	Specific enthalpy of sugar liquor, in
HF_o	STV	MJ/kg	Specific enthalpy of sugar liquor, out
x_i	INP	kg/kg	Solute mass fraction, in
x_o	STV	kg/kg	Solute mass fraction, out
TF_i	INP	C	Temperature of sugar liquor, in
TF_o	ALG	C	Temperature of sugar liquor, out
TFM	ALG	C	Weighted mean temperature of liquor
WG_i	ALG	kg/s	Steam flow, in
PG	INP	bar	Steam pressure
WC	ALG	kg/s	Flow of condensate, out
HC	ALG	MJ/kg	Specific enthalpy of condensate
TT	STV	C	Tube temperature
FVOL	PAR	m3	Volume of sugar liquor in the tubes
CT	PAR	MJ/K	Heat capacity of the tubes
k_{GT}	PAR	MW/K	Heat transfer coeff., steam to tubes
k_{TF}	PAR	MW/K	Heat transfer coeff., tubes to liquor

2.7. Model

It is assumed that the steam is saturated

$$TG = TSAF(PG)$$
$$HG_o = HGSF(TG)/1000.$$
$$HC = HFSF(TG)/1000.$$

where HG_o is the specific enthalpy of the steam. The heat flow and the flow of condensate is calculated

$$QGT = kGT \cdot (TG-TT)$$

$$WC = \frac{QGT}{HG_o - HC}$$

Applying conservation of mass the inlet flow of steam is

$$WG_i = WC$$

Using sugar functions the temperature and the specific heat capacity of the sugar liquor can be calculated.

$$TF_o = TSAPTH(HF_o, x_o)$$
$$c_{pf} = CPSAPTH(HF_o, x_o)$$

The total heat transfer coefficient UA is given by

$$\frac{1}{UA} = \frac{1}{k_{GT}} + \frac{1}{k_{TP}}$$

The function TMA gives weight factor B

$$B = TMA(UA, WF_i \cdot c_{pf})$$

which is being used for calculation of the mean temperature TFM of the sugar liquor

$$TFM = B \cdot TF_i + (1-B) \cdot TF_o$$

Now the heat flow from the tubes to the liquor can be found

$$QTF = k_{TP} \cdot (TT-TFM)$$

The mass of sugar liquor is calculated by first finding the density:

$$\rho = RFSP(TFM)$$
$$Mass = FVOL \cdot \rho$$

Conservation of mass applied on the sugar in the tubes leads to

$$\dot{x}_o = \frac{WF_i \cdot (x_i - x_o)}{Mass}$$

because the outlet flow is the same as the inlet flow. The same argument and application of energy conservation on the sugar liquor gives

$$\dot{HF}_o = \frac{WF_i \cdot (HF_i - HF_o) + QTF}{Mass}$$

and application of energy conservation on the tubes yields

$$\dot{TT} = \frac{QGT - QTF}{CT}$$

3. ROBERT EVAPORATOR

There are two different models for a Robert evaporator. The first one, ROBERT, can be used in models where the pressure dynamics is included while the other, ROQSTA, can be used in models where a quasi-stationary expression is being used for the pressure calculation.

3.1.1. Name

ROBERT

3.1.2. Description

A flow of water with sugar, sugar liquor, is entering the evaporator at the bottom volume (see Fig. F1). From there it can enter a section with tubes. On the outside of the tubes steam is being condensed and the heat of condensation is used for evaporation of water from the sugar liquor. When the water evaporates some of the sugar liquor is dragged up out of the tubes and some of it drops into a center tube from where it leaves the evaporator.

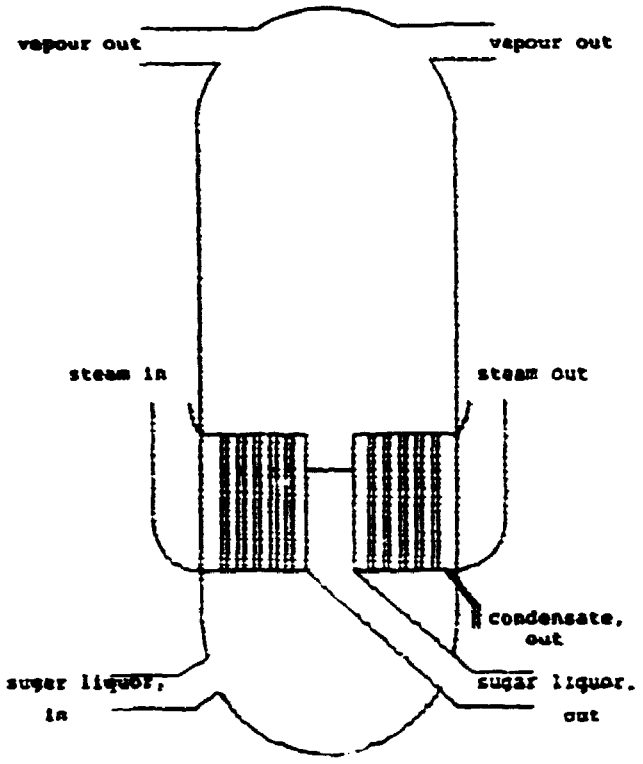


Fig. P1. Schematic presentation of a Robert evaporator.

The condensed steam flows out of the evaporator, and also some steam can be drawn from the other side.

The amount of steam evaporated from the sugar liquor is determined from the degree of superheating. The pressure of the vapour above the sugar liquor is calculated dynamically by lumping this volume together with other volumes at approximately the same pressure and applying conservation of mass. These volumes will be in components to which the connecting tubes have no valves. The reason for not making a dynamic pressure calculation in the evaporator is that it may introduce a very stiff system of differential equations.

It will be assumed that the sugar liquor is well stirred so there are no gradients of temperature, concentration, etc. in the liquor. Furthermore it will be assumed that there is a well defined level of sugar liquor. This level can be used for control of the outlet flow of sugar liquor.

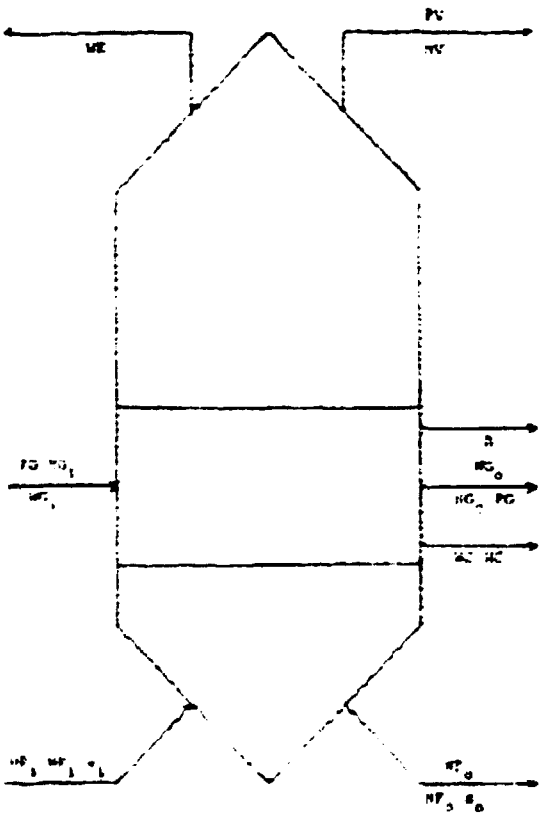
3.1.3. Characteristics

This evaporator with external dynamic pressure calculation may result in a stiff system and should be used only in models where pressure dynamics is important. The model of the sugar liquor is rather simple as it is lumped to only one lump and a well-defined level is being assumed. The heat transfer resistance over the tubes is approximated by two resistances on the outside and on the inside of the tubes. It is split equally.

3.1.4. Prerequisites

The library H2OFUNCTIONS must be available. The sugar functions EBP and HSAFTT must be available and the function UAFH for calculation of the heat transfer efficiency dependence on sugar level must be available.

3.1.5. Symbols



3.1.6. Identifiers

Name	Type	Dimension	Description
WP_i	INP	kg/s	Inlet flow of sugar liquor
WP_o	INP	kg/s	Outlet flow of sugar liquor
HP_i	INP	MJ/kg	Specific enthalpy of sugar liquor, in
HP_o	ALG	MJ/kg	Specific enthalpy of sugar liquor, out
x_i	INP	kg/kg	Solute mass fraction, in
x_o	STV	kg/kg	Solute mass fraction, out
PG	INP	bar	Steam pressure
PV	INP	bar	Vapour pressure
HG_o	ALG	MJ/kg	Specific enthalpy of steam, out
HC	ALG	MJ/kg	Specific enthalpy of condensate
HV	ALG	MJ/kg	Specific enthalpy of vapour
WG_i	ALG	kg/s	Steam flow, in
WG_o	INP	kg/s	Steam flow, out
WC	ALG	kg/s	Condensate flow
WE	ALG	kg/s	Evaporation rate
TP	STV	C	Temperature of sugar liquor
TT	STV	C	Temperature of tubes
h	STV	m	Level of sugar liquor
h_{opt}	PAR	m	Optimal height for heat transfer
A	PAR	m^2	Surface area of sugar liquor
CT	PAR	MJ/C	Heat capacity of tubes
AUMAX	PAR	MW/C	Maximal heat transfer coefficient
V_u	PAR	m^3	Volume of sugar liquor under the tubes

3.1.7. Model

The function UAFH gives the heat transfer efficiency as a function of both the level and optimal level. This fraction multiplied on the maximal heat transfer coefficient gives the actual coefficient. The heat transfer resistance on the inside and outside of the tubes and across the tube is split into two resistances, one on the inside and the other on the outside of the tubes in equal proportions. The the heat transfer coefficients can now be calculated

k_{GT} = UAFH(h,h_{opt})*AUMAX*2

k_{TF} = k_{GT}

Since the steam is condensing the steam temperature is found using the saturation temperature

TG = TSAF(PG)

and now the heat flow from the steam to the tubes and from the tubes to the sugar liquor can be calculated

QGT = k_{GT}*(TG-TT)

QTF = k_{TF}*(TT-TF)

Again because of saturation the specific enthalpies of condensate and of steam can be found

HC = HFSF(TG)/1000.

HG_O = HGSF(TG)/1000.

and the flow of condensate can be found by applying conservation of energy

$$WC = \frac{QGT}{HG_O - HC}$$

Using mass conservation, the inlet flow of steam is

WG_i = WG_O+WC

The sugar function EBP gives the elevation of the boiling point of sugar liquor due to the sugar

DT = EBP(TF,x_O)

The vapour is at saturation and therefore the temperature and specific enthalpy can be found

TV = TSAF(PV)

HV = HGSF(TV)/1000.

The degree of superheating is found by assuming that the water is evaporating from pure water at the temperature of the sugar liquor reduced by the elevation of the boiling point. The degree of superheating is then

$$T_{super} = (TF-DT)-TV$$

The mass of the sugar liquor is calculated by assuming that the density of the liquor is the same as that of pure water.

$$\rho = RFSF(TF)$$
$$Mass = (A \cdot h + V_u) \cdot \rho$$

The evaporation rate is approximated by

$$WE = \begin{cases} 0.008 \cdot Mass \cdot T_{super}^2 & T_{super} > 0 \\ 0 & T_{super} < 0 \end{cases}$$

The specific enthalpy of the sugar liquor is calculated by using a sugar function

$$HF_o = HSAPTT(TF, x_o)$$

Applying conservation of mass on the sugar liquor gives

$$\dot{h} = \frac{WF_i - (WE + WF_o)}{A \cdot \rho}$$

By denoting the mass of sugar in the sugar liquor M_t the following expression can be written

$$M_t = (V_u + A \cdot h) \cdot \rho \cdot x_o$$

which by differentiation gives

$$\dot{M}_t = (A \cdot \rho \cdot \dot{h} \cdot x_o + Mass \cdot \dot{x}_o)$$

$\dot{M}_t$ is the accumulation rate and applying mass conservation of

sugar on the sugar liquor the following expression is found

$$\dot{M}_t = WF_i \cdot x_i - WF_o \cdot x_o$$

Manipulation of these expressions for $\dot{M}_t$ gives

$$\dot{x}_o = \frac{WF_i \cdot x_i - WF_o \cdot x_o - A \cdot \rho \cdot \dot{h} \cdot x_o}{\text{Mass}}$$

Using the expression for $\dot{h}$ one gets after some manipulations

$$\dot{x}_o = \frac{WF_i \cdot (x_i - x_o) + W_e \cdot x_o}{\text{Mass}}$$

Derivation of the expression for the temperature of the sugar liquor follows the same line. Assuming that $h = c_p \cdot T$ where c_p is the specific heat capacity the time derivative of the enthalpy of the sugar liquor E is

$$\dot{E} = c_p \cdot \rho \cdot (\dot{V} \cdot TF + V \cdot \dot{TF})$$

where V is the total volume of sugar liquor. Applying energy conservation the accumulation of energy can be found

$$\dot{E} = WF_i \cdot HF_i + QTF - WE \cdot HV - WF_o \cdot HF_o$$

Manipulations of these two expressions give

$$\dot{T}_f = \frac{WF_i \cdot HF_i + QTF - WE \cdot HV - WF_o \cdot HF_o}{c_p \cdot \rho \cdot V} - T_f \frac{\dot{V}}{V} \frac{c_p \cdot \rho}{c_p \cdot \rho}$$

Using

$$\dot{V} = A \cdot \dot{h}$$

and the expression for $\dot{h}$ some further manipulations give

$$\dot{TF} = \frac{WF_i \cdot HF_i + QTF - WE \cdot HV - WF_o \cdot HF_o - (A \cdot \rho \cdot \dot{h}) \cdot c_p \cdot TF}{Mass \cdot c_p}$$

By the assumption above

$$\dot{hf}_o = c_p \cdot \dot{TF}$$

which upon insertion and reorganization gives

$$\dot{TF} = \frac{WF_i \cdot (HF_i - HF_o) - WE \cdot (HV - HF_o) + QTF}{Mass \cdot c_p}$$

Conservation of energy applied to the tubes gives

$$\dot{TT} = \frac{QGT - QTF}{CT}$$

3.2.1. Name

ROQSTA

3.2.2. Description

A flow of water containing sugar, sugar liquor, enters the evaporator at the bottom volume (see Fig. F1). From there it can enter a section with tubes. On the outside of the tubes steam is condensed and the heat of condensation is used for evaporation of water from the sugar liquor. When the water evaporates some of the sugar liquor is dragged up out of the tubes and some of it drops into a center tube from where it leaves the evaporator.

The steam that is condensed flows out of the evaporator and also some steam can be drawn from the other side.

The pressure calculation is made by assuming that the vapour above the liquor is at saturation. In this way a stiff system of differential equations is avoided by using this model.

It will be assumed that the sugar liquor is well stirred so there are no gradients of temperature, concentration, etc. in the liquor. Furthermore, it will be assumed that there is a well-defined level of sugar liquor. This level can be used for control of the outlet flow of sugar liquor.

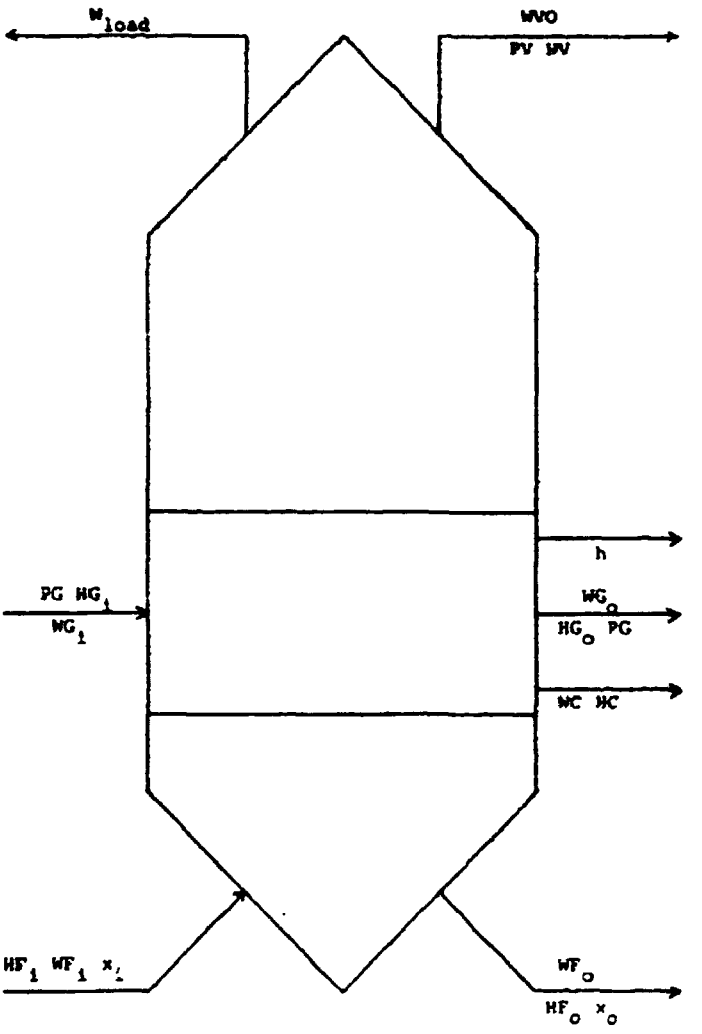
3.2.3. Characteristics

This model will not introduce a stiff system of differential equations due to dynamic pressure calculations. The model of the sugar liquor is rather simple as it is lumped to only one lump and a well-defined level is being assumed. The heat transfer resistance over the tubes is approximated by two resistances on the outside and on the inside of the tubes. It is split equally.

3.2.4. Prerequisites

The library H2OFUNCTIONS must be available. The sugar functions EBP and HSAFTT must be available and the function UAFH for calculation of the heat transfer efficiency dependence on sugar level must be available.

3.2.5. Symbols



3.2.6. Identifiers

Name	Type	Dimension	Description
WF_i	INP	kg/s	Inlet flow of sugar liquor
WF_o	INP	kg/s	Outlet flow of sugar liquor
HF_i	INP	MJ/kg	Specific enthalpy of sugar liquor, in
HF_o	ALG	MJ/kg	Specific enthalpy of sugar liquor, out
x_i	INP	kg/kg	Solute mass fraction, in
x_o	STV	kg/kg	Solute mass fraction, out
PG	INP	bar	Steam pressure
PV	ALG	bar	Vapour pressure

HG_o	ALG	MJ/kg	Specific enthalpy of steam, out
HC	ALG	MJ/kg	Specific enthalpy of condensate
HV	ALG	MJ/kg	Specific enthalpy of vapour
WG_i	ALG	kg/s	Steam flow, in
WG_o	INP	kg/s	Steam flow, out
WC	ALG	kg/s	Condensate flow
W_{load}	INP	kg/s	Extraction of vapour, external
WV_o	INP	kg/s	Extraction of vapour, internal
TF	STV	C	Temperature of sugar liquor
TT	STV	C	Temperature of tubes
h	STV	m	Level of sugar liquor
h_{opt}	PAR	m	Optimal height for heat transfer
A	PAR	m2	Surface area of sugar liquor
CT	PAR	MJ/C	Heat capacity of tubes
AUMAX	PAR	MW/C	Maximal heat transfer coefficient
V_u	PAR	m3	Volume of sugar liquor under the tubes

3.2.7. Model

This model differs only from the model of the evaporator prepared for dynamic calculations of vapour pressure in a few expressions. Where the expressions are the same they will be copied from 3.1.7 in this section.

$$k_{GT} = UAFH(h, h_{opt}) \cdot AUMAX \cdot 2$$

$$k_{TF} = k_{GT}$$

$$TG = TSAF(PG)$$

$$QGT = k_{GT} \cdot (TG - TT)$$

$$QTF = k_{TF} \cdot (TT - TF)$$

$$HC = HFSF(TG)/1000.$$

$$HG_o = HGSF(TG)/1000.$$

$$WC = \frac{QGT}{HG_o - HC}$$

$$WG_i = WG_o + WC$$

$$DT = EPB(TF, x_o)$$

The vapour is assumed to be at saturation above pure water at a temperature which is that of the sugar liquor but reduced with the elevation of boiling point

$$TV = TF-DT$$

$$HV = HFSF(TV)/1000.$$

$$\rho = RFSF(TF)$$

$$Mass = (A \cdot h+V_u) \cdot \rho$$

$$HF_o = HSAPTT(TF, x_o)$$

$$WE = W_{load} + WV_o$$

$$\dot{h} = \frac{WF_i-(WE+WF_o)}{A}$$

$$\dot{x}o = \frac{WF_i \cdot (x_i-x_o)+WE \cdot x_o}{Mass}$$

$$\dot{TF} = \frac{WF_i \cdot (HF_i-HF_o)-WE \cdot (HV-HF_o)+QTF}{Mass \cdot c_p}$$

$$\dot{TT} = \frac{QGT-QTF}{CT}$$

$$PV = PSAF(TV)$$

4. RAMP SWITCH

4.1. Name

RAMPSW

4.2. Description

This switch is designed to be used when a ramp must change a value of a variable. The ramp is programmed with a set of par-

ameters. The switch is activated by changing the values of the set of parameters from one state to another and the switch is activated when a signal crosses some levels except if the set of parameters already has the right set of values. The time step is controlled to activate the switch at the signal levels within a given accuracy.

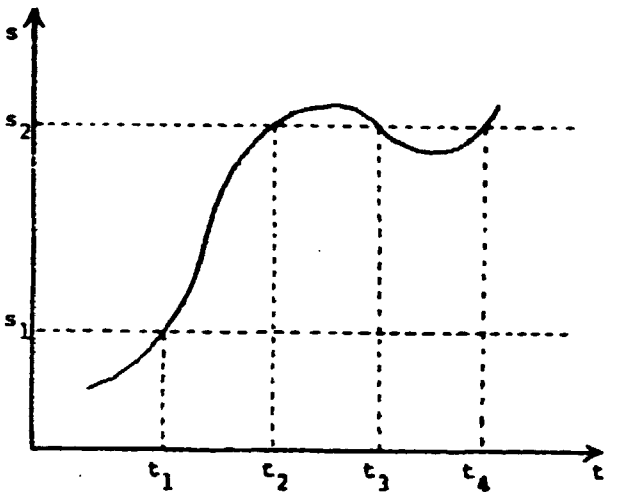


Fig. F2. Example of a switch event.

There are two levels for the signal (see Fig. F2) and two sets of values for the set of parameters. The description of the ramp switch is going to be made by looking at an example where the set of parameters initially is in state 1. When the signal crosses the value s_1 at time t_1 nothing happens because entering the center interval $[s_1,s_2]$ does not give rise to any changes. At t_2 the signal leaves this interval and enters one where the set of parameters must switch to state 2. At time t_3 the signal again enters the center interval and no change occurs. When the signal at t_4 leaves the center interval and enters the interval where the set of parameters must be at state 2 the switch will not be activated because it is already in this state.

This switch is programmed to fit the runtime executive DYSIM86 (CHRISTENSEN, KOFOED and LARSEN, 1986) since the integer NR is being used to consider only a switch at the last call of a time step. The size of the time step can be reduced by using the

REPET feature of DYSIM86 which also handles the insertion of the old values for output variables if a time step is rejected. The switch cannot be activated in a steady-state simulation.

4.3. Characteristics

The set of parameters that can be used to programme the switch constitutes of the times at which to initiate and terminate the ramp, TI and TT, and the initial and the final value, PI and PT, of the variable that is going to be changed between state 1 and 2 or vice versa. The ramp can then, for example, be programmed by using the following expression

```
P = PI+AMIN1(1.,(T-TI)/(TT-TI))*(PT-PI)
```

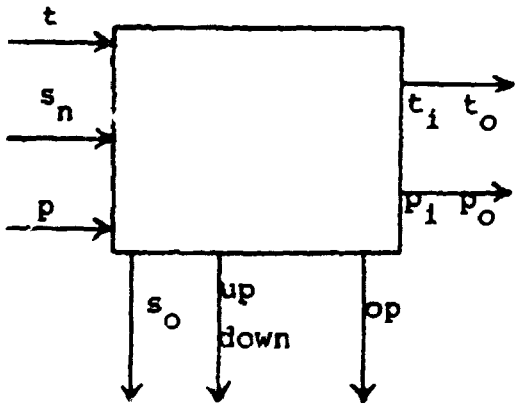
where

P is the variable following the ramp
T is the time
AMIN1 is a FORTRAN function returning the value of the least of the arguments

4.4. Prerequisites

None

4.5. Symbols



4.6. Identifiers

Name	Type	Dimension	Description
t	INP	s	Time
s_n	INP	anything	Present signal
p	INP	anything	Present value of the variable
t_i	ALG	s	Initiation time for ramp, if activated
t_t	ALG	s	Termination time for ramp, if activated
p_i	ALG	anything	Initial value for variable in ramp
p_t	ALG	anything	Final value for variable in ramp
s_o	ALG	anything	Old value of signal
op	ALG		Old parameter set, 1 or 2
c_1	ALG		Number of times switching to state 1
c_2	ALG		Number of times switching to state 2
s_1	PAR	anything	Lower level for switch
s_2	PAR	anything	Upper level for switch
ds	PAR	anything	Accuracy for state event
p_1	PAR	anything	Value of variable in state 1
p_2	PAR	anything	Value of variable in state 2
dpdt	PAR	anything	Rate of change of variable in lin. ramp

4.7. Model

The switch can only be activated in transients and only at the final calculation in a time step because at this time the value of the state variables, which is going to determine whether a state event has occurred, has got the values that are going to be used at the beginning of the next time step, that is if this time step is not rejected. The time step can be rejected because the state event is not detected with adequate accuracy.

A state event is detected if the new value of the signal is outside the center interval $[s_1,s_2]$ and if the old value was not also outside the center interval and to the same side.

In this case the switch can be considered only if the set of parameters is not already in the correct state. If it is not in

the right state it must be checked whether the state event has been detected within the desired accuracy which means that the absolute difference between the actual state and the signal value being crossed does not exceed ds. If it does exceed ds the time step must be reduced in order to detect the state event with greater accuracy. This is accomplished by calling a subroutine, REPET, in the runtime executive DYSIM86 (CHRISTENSEN, KOPCED and LARSEN, 1986).

The ramp switch is being activated by changing the set of parameters in such a way that a linear ramp is starting to change the value of the variable immediately, i.e. t_i is equal to the present time t. The start value p_i for the variable in the ramp is the present value p and the final value is either state 1 or state 2, p_1 or p_2. The termination time for the ramp is calculated by letting the variable value change with a constant rate independent of the amount that the variable has to be changed. There is also a variable that indicates the present state of the variable, i.e. which state it is at or is approaching. Also two counters have been made part of the model. They keep track of the number of times the state has been switched to state 1 and to state 2. The following equations describe the model in the case when a state event has been detected with adequate accuracy.

$$p_i = p$$
$$p_t = p_1 \ (p_2)$$
$$t_i = t$$
$$t_t = t_i + ABS(p_t - p_i)/dpdt$$
$$op = 1 \quad (2)$$
$$c_1 \ (c_2) = c_1(c_2) + 1$$

5. P-CONTROLLER

5.1. Name

PCON

5.2. Description

The P-controller output has an offset. The output differs from this offset with an amount that is proportional to the difference between the two input signals. The offset should be close to the working point because in this case the P-controller can make the difference between the input signals vanish.

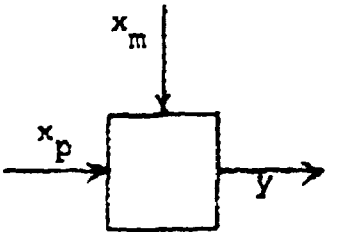
5.3. Characteristics

The input signals are limited to a measuring range and the output signal is limited to an output range.

5.4. Prerequisites

The function ALIM that limits the values of a variable to a given range must be available.

5.5. Symbols



5.6. Identifiers

Name	Type	Dimension	Description
x_p	INP	anything	(Positive) input
x_m	INP	anything	(Negative) input
x_{min}	PAR	anything	Lower limit in measuring range
x_{max}	PAR	anything	Upper limit in measuring range
P_b	PAR		Proportional band
y_{min}	PAR		Lower limit in output range
y_{max}	PAR		Upper limit in output range
y_s	PAR		Offset for output
y	ALG		Output

5.7. Model

The input range and the limitation of the input signals are calculated

$$RANGE = x_{max} - x_{min}$$
$$u = ALIM(x_{min}, x_p, x_{max})$$
$$v = ALIM(x_{min}, x_m, x_{max})$$

The unit error signal is

$$\varepsilon = \frac{u-v}{RANGE}$$

and the output is calculated and subjected to the limits

$$y = ALIM(y_{min}, (\varepsilon/P_b + y_s), y_{max})$$

6. PI-CONTROLLER

6.1. Name

PICON

6.2. Description

The PI-controller output is proportional to the sum of the error and integrated error where the error is the difference between the two input signals.

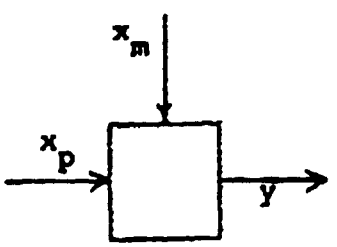
6.3. Characteristics

The input signals are limited to a measuring range and the output signal is limited to an output range. If the output is limited by this model so is the value of the integrated error as well as the derivative of this state variable.

6.4. Prerequisites

The function ALIM that limits the values of a variable to a given range and the function ALIMD that limits the derivative of a state variable must be available.

6.5. Symbols



6.6. Identifiers

Name	Type	Dimension	Description
x_p	INP	anything	(Positive) input
x_m	INP	anything	(Negative) input
x_{min}	PAR	anything	Lower limit in measuring range
x_{max}	PAR	anything	Upper limit in measuring range
z	STV	anything	Integrated error (proportional to)
τ	PAR	s	Reset time
P_b	PAR		Proportional band
y_{min}	PAR		Lower limit in output range
y_{max}	PAR		Upper limit in output range
y	ALG		Output

6.7. Model

The input range and the limitation of the input signals are calculated

$$\begin{aligned} \text{RANGE} &= x_{max} - x_{min} \\ u &= \text{ALIM}(x_{min}, x_p, x_{max}) \\ v &= \text{ALIM}(x_{min}, x_m, x_{max}) \end{aligned}$$

The unit error signal is

$$e = \frac{u-v}{\text{RANGE}}$$

The following auxiliary variable is defined

$$\varepsilon = \frac{e}{P_b}$$

and then the following expression for the state variable that is related to the integrated error is written

$$\dot{z} = \varepsilon/\tau$$

z is limited to the interval $[y_{min}, y_{max}]$ and if z is in either limit and the sign of the derivative is such that the integration routine will make z leave the interval then the value of the derivative is made equal to zero.

The output of the PI-controller is

$$y = z+\varepsilon$$

The output y is limited to the interval $[y_{min}, y_{max}]$. When the controller has succeeded in making the two input signals equal ε becomes zero and y = z. Therefore z is subjected to the same limits as y.

7. DELIMITER FUNCTIONS

The controllers may use these functions to restrict the input signals to the measuring range and the output signals to the output range. The function ALIM restricts the value to a given range and ALIMD changes the value of the corresponding derivative to zero if the state variable has been limited to the output range. Examples of the use of the functions is found in the submodule PICON (see Appendix G). These functions return values of the type REAL.

7.1. ALIM

The argument list is

$$(y_{min}, Y, y_{max})$$

where $[y_{min}, y_{max}]$ is the range that the values of y is restricted to. The values in the argument list must be REAL.

The function returns the value y if it is in the range $[y_{min}, y_{max}]$. If y is outside the range the function returns the value in the range that is closest to y.

The function is programmed as follows

ALIM = AMAX1(ymin,AMIN1(y,ymax))

and it can be seen that it is possible to use a statement function instead of the library. Statement functions are executed faster than ordinary functions.

7.2. ALIMD

The argument list is

(x,ymin,y,ymax)

where $[y_{min}, y_{max}]$ is the range that the values of the state variable y is restricted to and x is the value of the derivative. The values in the argument list must be REAL.

If the user has limited a state variable with for example ALIM it would be proper to also change the value of the derivative, at least in some cases. If the state variable has a value that is equal to, for example, the upper limit in the range, and the value of the derivative is positive, then the integration routine will change the value of the state variable to one outside the range. This is undesirable and therefore the function ALIMD should be used when calculating the derivative.

If the value of y is on the boundary of the range $[y_{min}, y_{max}]$ and the value of x has a sign such that the integration routine will make y leave this range then the function will return the value zero. In any other case the function returns the value x.

8. WEIGHT FACTOR FOR CALCULATION OF THE MEAN TEMPERATURE

The function TMA returns a value of the type REAL. The argument list is

(X,W)

where X is the heat transfer rate across the boundary separating two compartments and W is a measure of a heat transport rate through a compartment. These values must be REAL.

When the temperature of water flowing through a heat exchanger (fluid to fluid), or a preheater (gas to fluid), the component should be modelled as a distributed system to get a model that is "accurate". In static situations the model can be solved analytically by solving a coupled set of partial differential equations. This makes it possible to find the mean temperature as a weighted sum of the inlet and outlet temperatures. The idea then is to use this weight factor to calculate the mean temperature in a dynamic simulation also. This mean temperature is used in the calculation of the heat transfer rate.

The function depends on the ratio z of the heat transfer rate across the boundary and the heat transport through the tube. The heat transfer rate is kL while the heat transport rate is either

$$W \cdot c_p \qquad \text{(preheater)}$$

$$\left(\frac{1}{W_1 c_{p1}} - \frac{1}{W_2 c_{p2}}\right)^{-1} \qquad \text{(heat exchanger)}$$

In the preheater case z is calculated as

$$z = \frac{kL}{W \cdot c_p}$$

The weight factor B is then given by

$$B = \frac{1}{z} - \frac{\exp(-z)}{1-\exp(-z)}.$$

The derivation of these expressions as well as of the approximations for small and large values of z is given elsewhere (KOFOED, 1986).

9. HEAT TRANSFER EFFICIENCY ON A ROBERT EVAPORATOR, UAFH

The function UAFH returns a value of the type REAL. The argument list is

(h, h_{opt})

where h is the sugar liquor level and h_{opt} is the optimal level for heat transfer.

The heat transfer in a Robert evaporator depends on several parameters including the solute mass fraction, sugar liquor flow rate, temperature difference and sugar liquor level (JØRGENSEN, 1986). From these data the dependence on the level for a specific set of tests performed on the multiple effect evaporator being modelled has been approximated.

The efficiency has a maximum (=1) at the level h_{opt}. On either side of this level the efficiency drops linearly, most steeply for levels below hopt. For the actual expression the reader is referred to Appendix H with a listing of the library source file.

10. SUGAR FUNCTIONS

Four sugar functions were needed in the study of the multiple effect evaporator. These are all of the type REAL. The values in their argument list are of type REAL as well.

10.1. Elevation of boiling point, EBP

The content of sugar causes an elevation of the boiling point of the sugar liquor. The argument list is

(T,x)

where T is the temperature and x is the solute mass fraction. The expression for EBP was taken from (JØRGENSEN, 1985), notice the difference between the unit used for the solute mass fraction in the two works: B=100•x.

The function is a rather complicated expression involving the exponential function. For the actual expression the reader is referred to Appendix H with a listing of the library source file.

10.2. Specific enthalpy, HSAFTT

The argument list is

(T,x)

where T is the temperature and x the solute mass fraction. The expression for HSAFTT was taken from (JØRGENSEN, 1985), notice the difference between the units used for both the specific enthalpy and solute mass fraction.

The expression being used for HSAFTT is

$$H = (A(x)\cdot T^2+B(x)\cdot T+1.5)/1000$$

where

$$A(x) = 5.5\cdot 10^{-3}+3.75\cdot 10^{-3}\cdot x$$
$$B(x) = 4.122-2.51\cdot x$$

10.3. Sugar liquor temperature, TSAFTH

In some of the calculations it is necessary to know the temperature of the sugar liquor in a model where the "primary" variable is the specific enthalpy. This is possible by finding the inverse of HSAFTT.

The argument list is

$$(T,x)$$

where T is the temperature and x is solute mass fraction.

Let $C(H) = 1.5-1000\cdot H$, then the following quadratic expression is found from Section 10.2.

$$A(x)\cdot T^2+B(x)\cdot T+C(H) = 0$$

which is solved for T

$$T = -\frac{B(x)}{2\,A(x)} \pm \sqrt{\left(\frac{B(x)}{2\,A(x)}\right)^2 - \frac{C(H)}{A(x)}}$$

This is the expression that is used for the calculation of TSAFTH.

10.4. Specific heat capacity, CPSAFT

The specific heat capacity cp is defined by (SMITH and VAN NESS,1975)

$$c_p = \frac{dH}{dT} \qquad \text{(constant pressure)}$$

i.e. an expression that can be derived from HSAFTT.

The argument list is

(T,x)

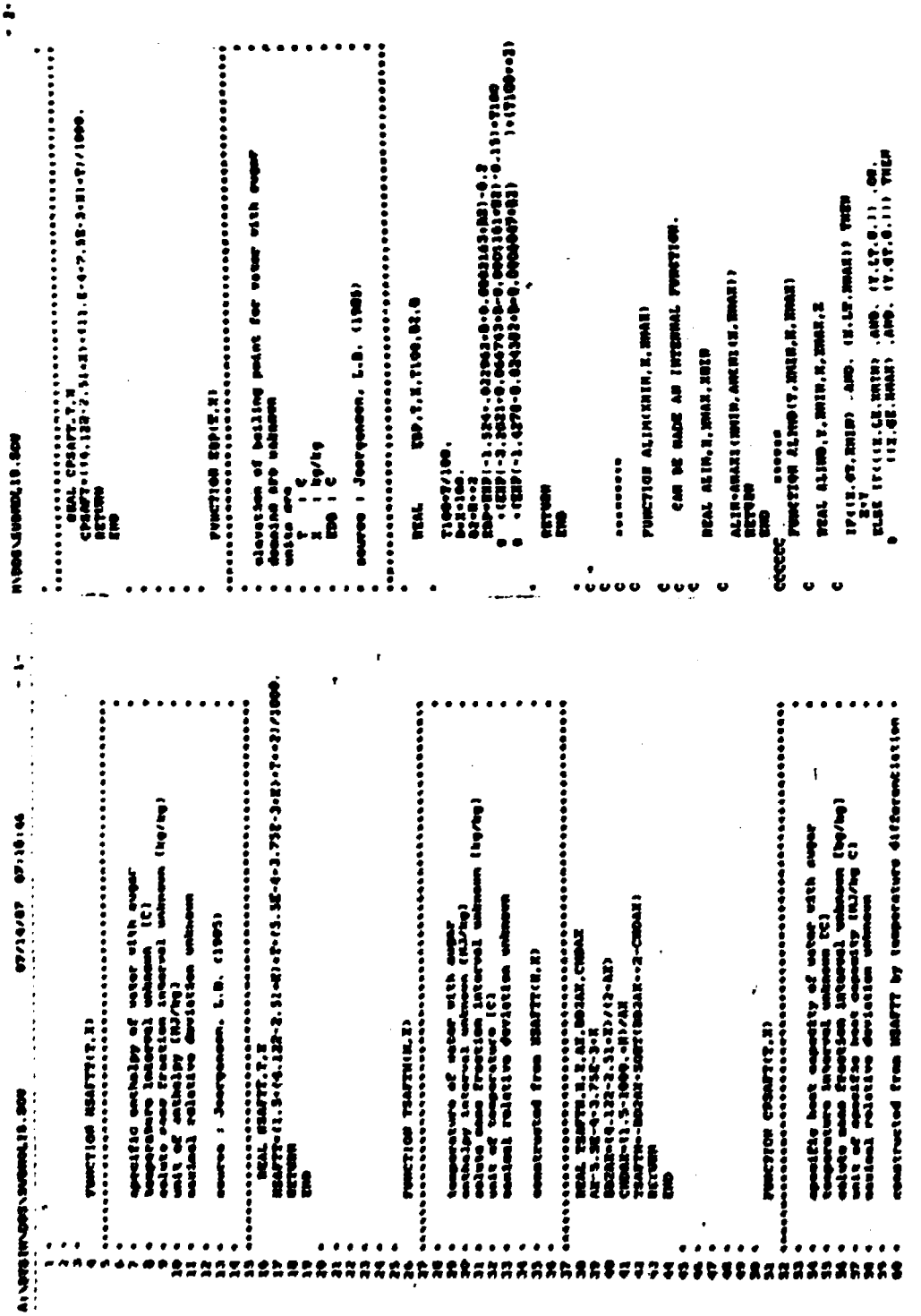
where T is the temperature and x is the solute mass fraction.

The expression becomes

```
CPSAFT = ((4.122-2.51*x)+(11.E-4+7.5E-3*x)*T)/1000.
```

APPENDIX G

LIBRARY WITH FUNCTIONS AND SUBMODULES USED FOR SIMULATION OF A SUGAR FACTORY



```
      Z=0.
    ELSE
      Z=Y
    END IF
    ALIMD=Z
C
    RETURN
    END
C
CCCCCC      ======
    FUNCTION UAFH(H,HOPT)
    REAL H,HOPT,ALFA,UAFH
    IF(H.LE.HOPT) THEN
      ALFA=2.831E-3
    ELSE
      ALFA=-79.7E-6
    END IF
    UAFH=1+(H-HOPT)*ALFA
    RETURN
    END
CCCCCC      ======
C
C
CCCCCC      ======
    FUNCTION TMA(X,W)
C   ABSOLUTE PRECISSION BETTER THAN 1.E-6
    REAL TMA,X,W,Z,B
    IF(ABS(W).GT.1.E-10*ABS(X)) THEN
      Z=X/W
      IF((1.E-3.LE.ABS(Z)) .AND. (ABS(Z).LE.1.E2)) THEN
        B=1./Z+1./(1.-EXP(Z))
      ELSE IF(ABS(Z).LE.1.E-3) THEN
        B=.5-Z/12.
      ELSE IF(1.E2.LE.Z) THEN
        B=1./Z
      ELSE IF(Z.LE.-1.E2) THEN
        B=1.+1./Z
      END IF
    ELSE IF(W.NE.0.) THEN
      B=1.-SIGN(1.,W)
    ELSE
C   W=0 IS NOT DEFINED
      B=-1.
    END IF
    TMA=B
    RETURN
    END
CCCCCC      ======
C
C
&   ------------------------------------------------------------
&   |                                                          |
&   |                                                          |
&   |     NOW THE SUBMODULES FOLLOW                            |
&   |                                                          |
&   |                                                          |
&   |                                                          |
&   ------------------------------------------------------------
```



```
&
&   ------------------------------------------------------------
C
!     SUBMO           PCON
&     =====           ====
&
&                P-CONTROLLER
&
!FUN    ALIM;
!INP    XP,XM;
!PAR    XMIN,XMAX,YMIN,YS,YMAX,PB;
!ALG Y;
&
&
&          XP         positive input
&          XM         negative  input
&
&          XMIN,XMAX  RANGE OF INPUT
&          YMIN,YMAX  RANGE OF OUTPUT
&          YS         offset for output
&          PB         PROPORTIONAL BAND
&
&          Y          OUTPUT SIGNAL
&
C
C
    REAL            EPSLON,RANGE,U,V,ALIM
C
C
    RANGE=XMAX-XMIN
    U=ALIM(XMIN,XP,XMAX)
    V=ALIM(XMIN,XM,XMAX)
    EPSLON=(U-V)/RANGE
    Y=ALIM(YMIN,(EPSLON/PB+YS),YMAX)
    RETURN
    END
&
&   ------------------------------------------------------------
C
!     SUBMO           PICON
&     =====           =====
!FUN    ALIM,ALIMD;
!STV    Z;
!INP    XP,XM;
!PAR    XMIN,XMAX,PB,TAU,YMIN,YMAX;
!ALG Y;
    REAL VP,VM,EPSLON,RANGE,ALIM,ALIMD
C
C   PI-CONTROLLER
C
    RANGE=XMAX-XMIN
    VP=ALIM(XMIN,XP,XMAX)
    VM=ALIM(XMIN,XM,XMAX)
    EPSLON=(VP-VM)/(PB*RANGE)
    Z=ALIM(YMIN,Z,YMAX)
    Z=ALIMD(EPSLON/TAU,YMIN,Z,YMAX)
    Y=ALIM(YMIN,EPSLON+Z,YMAX)
    RETURN
    END
&
```

```
241 &
242 &
243 &
244 &
245 *
246 *      SUBMO            ROBERT
247 &      ****             ******
248 &
249 &   pressure dynamics included in module
250 &
251 !FUN  HSAFTT,EBP,UAFH;
252 &
253 !STV  TF,TT,XO,H;
254 !ALG  WGI,WC,WE,HGO,HV,HC,HFO;
255 !INP  PG,PV,WGO,WFI,WFO,HFI,XI;
256 !PAR  A,CT,VU,AUMAX,HOPT;
257 &
258        REAL            TG,TV,TSUPER,DT,QGT,QTF,MASS,VOLUME,KGT,KTF,RHO
259          REAL     HSAFTT,EBP,UAFH
260 *
261        TG=TSAF(PG)
262        HGO=HGSF(TG)/1000.
263        HC =HFSF(TG)/1000.
264        KGT=UAFH(H,HOPT)*AUMAX*2.
265        KTF=KGT
266        QGT=KGT*(TG-TT)
267        WC =QGT/(HGO-HC)
268        WGI=WC+WGO
269        TV =TSAF(PV)
270        HV =HGSF(TV)/1000.
271        DT =EBP(TF,XO)
272        TSUPER=(TF-DT)-TV
273        VOLUME=A*H
274        RHO=RFSF(TF)
275        MASS=(VOLUME*VU)*RHO
276        IF(TSUPER.GE.0) THEN
277 *      SUPERHEATED FLUID
278          WE=0.008*MASS*TSUPER**2
279        ELSE
280          WE=0.
281        END IF
282        HFO=HSAFTT(TF,XO)
283        QTF=KTF*(TT-TF)
284        TF :(WFI*(HFI-HFO)+QTF-WE*(HV-HFO))/(MASS*(CPFSF(TF)/1000.))
285        XO :(WFI*(XI-XO)+WE*XO)/MASS
286        H  :(WFI-(WE+WFO))/(A*RHO)
287        TT :(QGT-QTF)/CT
288 *
289        RETURN
290        END
291 &
292 &      ------------------------------------------------------------------
293 *
294 !      SUBMO            ROQSTA
295 &      *****            ******
296 &
297 &   quasi stationary pressure calculation
298 &
299 !FUN  HSAFTT,EBP,UAFH;
300 &
```



```
!STV  TF,TT,XO,H;
!ALG  WGI,WC,TG,TV,PV,HGO,HV,HC,HFO;
!INP  PG,WGO,WFI,WFO,HFI,XI,WVO,WLOAD;
!PAR  A,CT,VU,AUMAX,HOPT;
&
       REAL            DT,QGT,QTF,MASS,VOLUME,KGT,KTF,WE,RHO
         REAL     HSAFTT,EBP,UAFH
*
       TG=AMAX1(TSAF(PG),TT)
       HGO=HGSF(TG)/1000.
       HC =HFSF(TG)/1000.
       KGT=UAFH(H,HOPT)*AUMAX*2.
       KTF=KGT
       QGT=KGT*(TG-TT)
       WC =QGT/(HGO-HC)
       WGI=WC+WGO
       WE=WLOAD+WVO
       DT =EBP(TF,XO)
       TV=TF-DT
       HV =HGSF(TV)/1000.
       PV=PSATF(TV)
       HFO=HSAFTT(TF,XO)
       VOLUME=A*H
       RHO=RFSF(TF)
       MASS=(VOLUME*VU)*RHO
       QTF=KTF*(TT-TF)
       TF :(WFI*(HFI-HFO)+QTF-WE*(HV-HFO))/(MASS*(CPFSF(TF)/1000.))
       XO :(WFI*(XI-XO)+WE*XO)/MASS
       H  :(WFI-(WE+WFO))/(A*RHO)
       TT :(QGT-QTF)/CT
*
       RETURN
       END
&
&      ------------------------------------------------------------------
&
&
&
!      SUBMO            VV
&      *****            **
&
! FUN TSAFTH,CPSAFT,TMA;
! STV XO,HFO,TT;
! ALG WGI,WC,HC,TFO,TFM;
! INP PG,WFI,HFI,XI,TFI;
! PAR FVOL,KGT,KTF,CT;
&
       REAL            TG,HG,QGT,QTF,MASS,UA,CPF,D
         REAL     TSAFTH,CPSAFT,TMA
*
       TG =AMAX1(TSAF(PG),TT)
       HG =HGSF(TG)/1000.
       HC =HFSF(TG)/1000.
       QGT=KGT*(TG-TT)
       WC =QGT/(HG-HC)
       WGI=WC
       TFO=TSAFTH(HFO,XO)
       MASS=FVOL*RFSF(TFM)
       XO :WFI*(XI-XO)/MASS
       UA=1./(1./KGT+1./KTF)
```

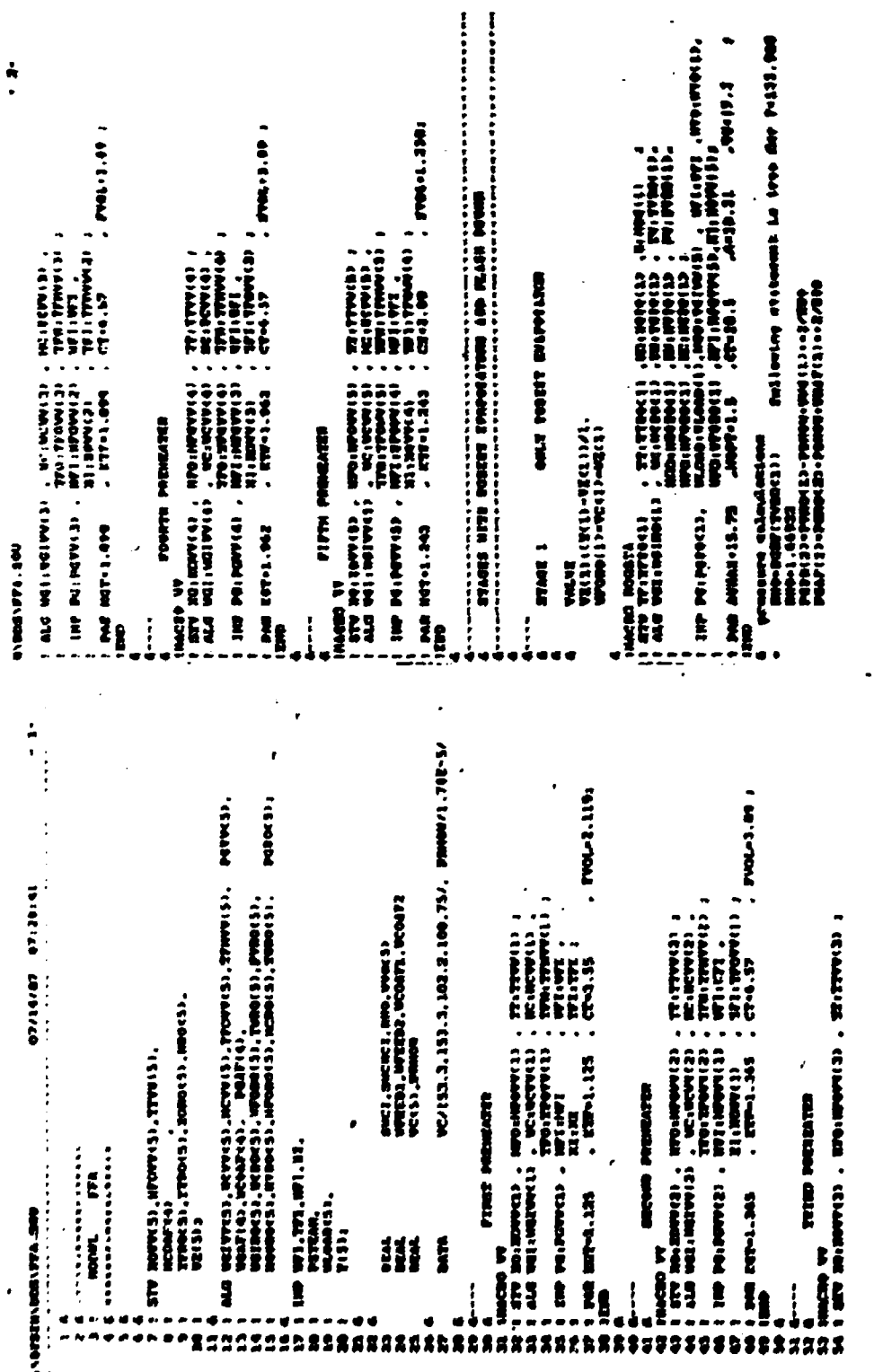
```
      CPF=CPSAFT(TFM,XO)
      B   =TMA(UA,WF1*CPF)
      TFM=B*TF1+(1.-B)*TF
      QTF=KTF*(TT-TFM)
      HFO=(WF1*(HF1-HFO)+QTF)/MASS
      TT  =(QGT-QTF)/CT
&
      RETURN
      END
&
&
&
&
&
!     SUBNO           AFSP
&     =====           ====
&
! STV                 MCO;
! ALG                 WG,WCO;
! INP                 SWCI,SWCHCI,PG;
! PAR                 V;
&
*
      REAL            TG,HG,TCO,TSUPER,M
*
      TG=TSAF(PG)
      M=V*RFSF(TG)
      HG=HGSF(TG)/1000.
      TCO=MCO/(CPFSF(TG)/1000.)
      TSUPER=TCO-TG
      IF(TSUPER.GT.0.) THEN
        WG=0.008*M*TSUPER**2
      ELSE
        WG=0.
      END IF
      WCO=SWCI-WG
      HCO=(SWCHCI-WCO*HCO-WG*HG)/M
*
      RETURN
      END
&
&     ------------------------------------------------------------------
&
&
&
!     SUBNO       RAMPSW
!ALG    SO,OP,C1,C2,TI,TT,PI,PT;
!INP    P,SN,T;
!PAR    DS,S1,S2,P1,P2,DPDT;
&
      IF(T.LT.0.) THEN
&     steady state
        CONTINUE
      ELSE
&     transient
        IF(NR.NE.4) THEN
&       not at the the end of a step
          CONTINUE
        ELSE
```



```
&         at the end of a step
&         switch should be considered if new state (SN) is not in contol
&         region : (S1,S2) _and_ old state was neither _and_ old state
&         was outside to the same side
          IF(.NOT. (((SN.LT.S1) .AND. .NOT.(SO.LT.S1)) .OR.
     *              ((SN.GT.S2) .AND. .NOT.(SO.GT.S2)))) THEN
            CONTINUE
          ELSE
&           a state event has occurred
            IF(SN.LT.S1) THEN
              IF(OP.EQ.1) THEN
&               old parameters already in state 1
                CONTINUE
              ELSE
                IF(SN.LT.S1-DS) THEN
                  CALL REPET
                ELSE
                  OP=1
                  C1=C1+1
                  P1=P
                  PT=P1
                  TI=T
                  TT=TI+ABS(PT-PI)/DPDT
                END IF
              END IF
            ELSE
&             SN>S2
              IF(OP.EQ.2) THEN
&               old parameters already in state 2
                CONTINUE
              ELSE
                IF(SN.GT.S2+DS) THEN
                  CALL REPET
                ELSE
                  OP=2
                  C2=C2+1
                  PI=P
                  PT=P2
                  TI=T
                  TT=TI+ABS(PT-PI)/DPDT
                END IF
              END IF
            END IF
          END IF
        END IF
&
        IF(NR.EQ.1) SO=SN
&
      END IF
&
      RETURN
      END
&     ------------------------------------------------------------------
&
```

APPENDIX H

LISTING OF SOURCE FILES THAT ARE BEING USED FOR SIMULATION OF A PART OF A SUGAR FACTORY



```
      PGVV(4)=PGRO(3)-PRHOV*WGIVV(4)**2/RHO
&
&
&     STAGE 3         FLASH DRUM AND ROBERT EVAPORATOR
&
      FLASH DRUM 1
      SWCI=WCRO(1)
      SWCHCI=WCRO(1)*HCRO(1)
!MACRO AFSP
! STV WCO:WCOAF(1) ;
! ALG WG:WGAF(1)    , WCO:WCOAF(1) ;
! INP SWCI:SWCI     , SWCHCI:SWCHCI , PG:PGAF(1) ;
! PAR V=1.8 ;
!END
&     calculate the amount of feedwater to be drawn from first stage
      WFEED1=AMIN1(WCOAF(1),WGIRO(1))
      WCOAF1=WCOAF(1)-WFEED1
&
&     VALVE
      V2(2)=(Y(2)-V2(2))/1.
      WFORO(2)=WC(2)*V2(2)
&
!MACRO ROOSTA
! STV TF:TFRO(2)    , TT:TTRO(2) , XO:XORO(2) , H:HRO(2) ;
! ALG WGI:WGIRO(2) , WC:WCRO(2) , TG:TGRO(2), TV:TVRO(2),
!                    WGO:WGORO(2) , HV:HVRO(2) ,PV:PVRO(2),
!                    WFO:WFORO(2) , HC:HCRO(2) ;
! INP PG:PGRO(2)    , WGO:WGIVV(4) , WFI:WFORO(1) ,WFO:WFORO(2) ,
!                 WFI:WFORO(1) , XI:XORO(1) ,WLOAD:WLOAD(2),WVO:WVO(2);
! PAR AUMAX=12.86    ,HOPT=1.5    ,CT=30.2     ,A=10.27     ,VU=17.2     ;
!END
&   pressure calculations
*     RHO=RGSF(TVRO(2))      following statement is true for T=126.102
      RHO=1.33969
      WVO(1)=WGIRO(2)-WGAF(1)
      PGRO(3)=PVRO(2)-PRHOV*WVO(2)**2/RHO
      PGAF(2)=PGRO(3)+PRHOV*WGAF(2)**2/RHO
      PGVV(3)=PGRO(3)-PRHOV*WGIVV(3)**2/RHO
&
&----
&     STAGE 3         FLASH DRUM AND ROBERT EVAPORATOR
&
&     FLASH DRUM 2
      SWCI=WCRO(2)+WCOAF1+WCVV(5)+WCVV(6)
      SWCHCI=WCRO(2)+HCRO(2)+WCOAF1+HCOAF(1)+WCVV(5)+HCVV(5)
     &                                        +WCVV(4)+HCVV(4)
!MACRO AFSP
! STV WCO:WCOAF(2) ;
! ALG WG:WGAF(2)    , WCO:WCOAF(2) ;
! INP SWCI:SWCI     , SWCHCI:SWCHCI , PG:PGAF(2) ;
! PAR V=1.8 ;
!END
&     calculate the amount of feedwater to be drawn from second stage
      WFEED2=WGIRO(1)-WFEED1
```



```
      WCOAF2=WCOAF(2)-WFEED2
&
&     VALVE
      V2(3)=(Y(3)-V2(3))/1.
      WFORO(3)=WC(3)*V2(3)
&
!MACRO ROOSTA
! STV TF:TFRO(3)    , TT:TTRO(3) , XO:XORO(3) , H:HRO(3) ;
! ALG WGI:WGIRO(3) , WC:WCRO(3) , TG:TGRO(3), TV:TVRO(3),
!                    WGO:WGORO(3) , HV:HVRO(3) ,PV:PVRO(3),
!                    WFO:WFORO(3) , HC:HCRO(3) ;
! INP PG:PGRO(3)    , WGO:WGIVV(3),WFI:WFORO(2) ,WLOAD:WLOAD(3),
!                 WFO:WFORO(3) ,WVO:WVO(3) , WFI:WFORO(2) , XI:XORO(2) ;
! PAR AUMAX=7.572    ,HOPT=1.5    ,CT=15.2     ,A=7.526     ,VU=10. ;
!END
&   pressure calculations
*     RHO=RGSF(TVRO(3))      following statement is true for T=116.619
      RHO=1.01821
      WVO(2)=WGIRO(3)-WGAF(2)
      PGRO(4)=PVRO(3)-PRHOV*WVO(3)**2/RHO
      PGAF(3)=PGRO(4)-PRHOV*WGAF(3)**2/RHO
      PGVV(2)=PGRO(4)-PRHOV*WGIVV(2)**2/RHO
&
&----
&     STAGE 4         FLASH DRUM AND ROBERT EVAPORATOR
&
&     FLASH DRUM 3
      SWCI=WCRO(3)+WCOAF2+WCVV(3)
      SWCHCI=WCRO(3)+HCRO(3)+WCOAF2+WCOAF(2)+WCVV(3)+HCVV(3)
!MACRO AFSP
! STV WCO:WCOAF(3) ;
! ALG WG:WGAF(3)    , WCO:WCOAF(3) ;
! INP SWCI:SWCI     , SWCHCI:SWCHCI , PG:PGAF(3) ;
! PAR V=1.8 ;
!END
&
&     VALVE
      V2(4)=(Y(4)-V2(4))/1.0
      WFORO(4)=WC(4)*V2(4)
&
!MACRO ROOSTA
! STV TF:TFRO(4)    , TT:TTRO(4) , XO:XORO(4) , H:HRO(4) ;
! ALG WGI:WGIRO(4) , WC:WCRO(4) , TG:TGRO(4), TV:TVRO(4),
!                    WGO:WGORO(4) , HV:HVRO(4) ,PV:PVRO(4),
!                    WFO:WFORO(4) , HC:HCRO(4) ;
! INP PG:PGRO(4)    , WGO:WGIVV(2),WFI:WFORO(3) ,WLOAD:WLOAD(4),
!                 WFO:WFORO(4) ,WFI:WFORO(3) , XI:XORO(3) ,WVO:WVO(4);
! PAR AUMAX=2.291    ,HOPT=1.5    ,CT=11.0     ,A=5.237     ,VU=6.4 ;
!END
&   pressure calculations
*     RHO=RGSF(TVRO(4))      following statement is true for T=104.817
      RHO=.700593
      WVO(3)=WGIRO(4)-WGAF(3)
      PGRO(5)=PVRO(4)-PRHOV*WVO(4)**2/RHO
```

```
      PGAF(4)=PGRO(5)+PRHOW*WGAF(4)**2/RHO
      PGVV(1)=PGRO(5)+PRHOW*WGIVV(1)**2/RHO
C
C
C     STAGE 1          FLASH DRUM AND ROBERT EVAPORATOR
C
C     FLASH DRUM 4
      WCI=WCRO(4)+WCRO(5)+WCOAF(3)+WCVV(2)+WCVV(1)
      WCHCI=WCRO(4)+WCRO(4)+WCRO(5)+HCRO(5)+WCOAF(3)+HCOAF(3)
     &                      +WCVV(2)+HCVV(2)+WCVV(1)+HCVV(1)
!MACRO AFSP
! STV HCO:HCOAF(4) ;
! ALG WG:WGAF(4)    , WCO:WCOAF(4) ;
! INP WCI:WCI       , WCHCI:WCHCI , PG:PGAF(4) ;
! PAR V=1.8 ;
!END
C
C     VALVE
      VZ(5)=(Y(5)-VZ(5))/1.
      WFORO(5)=VC(5)*VZ(5)
C
!MACRO ROQSTA
! STV TF:TFRO(5)    , TT:TTRO(5) , XO:XORO(5) , H:HRO(5) ;
! ALG WGI:WGIRO(5) , WC:WCRO(5) , TG:TGRO(5), TV:TVRO(5),
!                    WGO:WGORO(5) , HV:HVRO(5) ,PV:PVRO(5),
!                    WFO:WFORO(5) , HC:HCRO(5) ;
! INP PG:PGRO(5)    , WGO:WGIVV(1),WFI:WFORO(4) ,WLOAD:WLOAD(5),
!                WFO:WFORO(5) ,HFI:HFORO(4) , XI:XORO(4),WVO:0.;
! PAR AUMAX=0.6544  ,HOPT=1.5   ,CT=7.99    ,A=2.648    ,VU=3.2 ;
!END
C
      WVO(4)=WGIRO(5)-WGAF(4)
C
C
C-----------------------------------------------------------------------
C                    PRESSURE CALCULATION FOR STAGE 1
C-----------------------------------------------------------------------
C
      PGRO(1)=PSTEAM
*     RHO=RGSF(TSAF(PSTEAM)) following statement is true for T=140.346
      RHO=1.98609
      PGVV(5)=PSTEAM-PRHOW*WGIVV(5)**2/RHO
C
      RETURN
C
      ENTRY FFAOUT
C
      WRITE(10,*) ' Sugar liquor:'
      WRITE(10,9100) 'WFI (kg/s)','TFI (C)','HFI (MJ/kg)','XI (kg/kg)'
      WRITE(10,9200) WFI,TFI,HFI,XI
      WRITE(10,*) ' '
      WRITE(10,*) ' Steam:'
      WRITE(10,9300) 'P (bar)',PSTEAM
      WRITE(10,*) ' '
```



```
      WRITE(10,*) ' Load:'
      WRITE(10,9300) 'WLOAD (kg/s)',WLOAD
      WRITE(10,*) ' '
      WRITE(10,*) ' Preheaters:'
      WRITE(10,9300) 'XOVV (kg/kg)',XOVV
      WRITE(10,9300) 'HFOVV (MJ/kg)',HFOVV
      WRITE(10,9300) 'WGIVV (kg/s)',WGIVV
      WRITE(10,9300) 'WCVV (kg/s)',WCVV
      WRITE(10,9300) 'HCVV (MJ/kg)',HCVV
      WRITE(10,9300) 'TTVV (C)',TTVV
      WRITE(10,9300) 'TGVV (C)',(TSAT(PGVV(I)),I=1,5)
      WRITE(10,9300) 'TFOVV (C)',TFOVV
      WRITE(10,9300) 'TFMVV (C)',TFMVV
      WRITE(10,9300) 'PGVV (bar)',PGVV
      WRITE(10,*) ' '
      WRITE(10,*) ' Flash drums:'
      WRITE(10,9300) 'HCOAF (MJ/kg)',HCOAF
      WRITE(10,9300) 'WGAF (kg/s)',WGAF
      WRITE(10,9300) 'WCOAF (kg/s)',WCOAF
      WRITE(10,9300) 'PGAF (bar)',PGAF
      WRITE(10,9300) 'WFEED, 1 & 2',WFEED1,WFEED2
      WRITE(10,9300) 'WCOAF, 1 & 2',WCOAF1,WCOAF2
      WRITE(10,*) ' '
      WRITE(10,*) ' Controllers:'
      WRITE(10,9300) 'Y',Y
      WRITE(10,*) ' '
      WRITE(10,*) ' Robert evaporators:'
      WRITE(10,9300) 'TFRO (C)',TFRO
      WRITE(10,9300) 'TTRO (C)',TTRO
      WRITE(10,9300) 'TVRO (C)',TVRO
      WRITE(10,9300) 'TGRO (C)',TGRO
      WRITE(10,9300) 'XORO (kg/s)',XORO
      WRITE(10,9300) 'HRO (MJ/kg)',HRO
      WRITE(10,9300) 'WGIRO (kg/s)',WGIRO
      WRITE(10,9300) 'WCRO (kg/s)',WCRO
      WRITE(10,9300) 'WFORO (kg/s)',WFORO
      WRITE(10,9300) 'HGORO (MJ/kg)',HGORO
      WRITE(10,9300) 'HVRO (MJ/kg)',HVRO
      WRITE(10,9300) 'HFORO (MJ/kg)',HFORO
      WRITE(10,9300) 'HCRO (MJ/kg)',HCRO
      WRITE(10,9300) 'PGRO (bar)',PGRO
      WRITE(10,9300) 'PVRO (bar)',PVRO
      WRITE(10,*) ' '
C
      RETURN
 9100 FORMAT(1X,A10,5A12)
 9200 FORMAT(1X,6G12.5)
 9300 FORMAT(1X,A12,5G12.5)
C
      END
C
```

```
A:\DYSIM\DDS\REG.SOU                    07/14/87  07:24:36              - 1-
----------------------------------------------------------------------------
  1 !MODUL  REG
  2 !STV    Z(4),SZ;
  3 !ALG    Y(5),WLOAD(4),WFI,
  4 !       PVO(4),OP(4),DOWN(4),UP(4),TI(4),TT(4),PI(4),PT(4),WLOADS(4);
  5 !INP    WRO(5),WSET(5),WLOAD(4),PV(4);
  6 &
  7       REAL WAC
  8 &
  9 !MACRO      RAMPSW
 10 !ALG    SO:PVO(1), OP:OP(1), C1:DOWN(1), C2:UP(1),
 11 !       TI:TI(1) , TT:TT(1), PI:PI(1)    , PT:PT(1);
 12 !INP    P:WLOADS(1), SN:PV(1), T:T;
 13 !PAR    DS=.01, S1=1.5, S2=1.6, P1=0., P2=41.582, DPDT=1.00;
 14 !END
 15 !MACRO      RAMPSW
 16 !ALG    SO:PVO(2), OP:OP(2), C1:DOWN(2), C2:UP(2),
 17 !       TI:TI(2) , TT:TT(2), PI:PI(2)    , PT:PT(2);
 18 !INP    P:WLOADS(2), SN:PV(2), T:T;
 19 !PAR    DS=.01, S1=1.3, S2=1.4, P1=0., P2=41.212, DPDT=1.00;
 20 !END
 21 !MACRO      RAMPSW
 22 !ALG    SO:PVO(3), OP:OP(3), C1:DOWN(3), C2:UP(3),
 23 !       TI:TI(3) , TT:TT(3), PI:PI(3)    , PT:PT(3);
 24 !INP    P:WLOADS(3), SN:PV(3), T:T;
 25 !PAR    DS=.01, S1=0.3, S2=1.0, P1=0., P2=7.3362, DPDT=0.50;
 26 !END
 27 !MACRO      RAMPSW
 28 !ALG    SO:PVO(4), OP:OP(4), C1:DOWN(4), C2:UP(4),
 29 !       TI:TI(4) , TT:TT(4), PI:PI(4)    , PT:PT(4);
 30 !INP    P:WLOADS(4), SN:PV(4), T:T;
 31 !PAR    DS=.01, S1=0.2, S2=0.3, P1=0., P2=14.544, DPDT=0.50;
 32 !END
 33 &
 34       WLOADS(1)=PI(1)+AMIN1(1.,((T-TI(1))/(TT(1)-TI(1))))*(PT(1)-PI(1))
 35       WLOADS(2)=PI(2)+AMIN1(1.,((T-TI(2))/(TT(2)-TI(2))))*(PT(2)-PI(2))
 36       WLOADS(3)=PI(3)+AMIN1(1.,((T-TI(3))/(TT(3)-TI(3))))*(PT(3)-PI(3))
 37       WLOADS(4)=PI(4)+AMIN1(1.,((T-TI(4))/(TT(4)-TI(4))))*(PT(4)-PI(4))
 38 &
 39 !       MACRO PCON
 40 !       INP XP:WRO(1) , XM:WSET(1);
 41 !       PAR XMIN=1.25 , XMAX=1.75 , YMIN=0. , YMAX=1. ,YS=.72,PB=.030;
 42 !       ALG Y:Y(1);
 43 !       END
 44 !       MACRO PCON
 45 !       INP XP:WRO(2) , XM:WSET(2);
 46 !       PAR XMIN=1.25 , XMAX=1.75 , YMIN=0. , YMAX=1. , YS=.446,PB=.030;
 47 !       ALG Y:Y(2);
 48 !       END
 49 !       MACRO PCON
 50 !       INP XP:WRO(3) , XM:WSET(3);
 51 !       PAR XMIN=1.25 , XMAX=1.75 , YMIN=0. , YMAX=1. ,YS=.4249,PB=.027;
 52 !       ALG Y:Y(3);
 53 !       END
 54 !       MACRO PCON
 55 !       INP XP:WRO(4) , XM:WSET(4);
 56 !       PAR XMIN=1.25 , XMAX=1.75 , YMIN=0. , YMAX=1. ,YS=.3506,PB=.038;
 57 !       ALG Y:Y(4);
 58 !       END
 59 !       MACRO PCON
 60 !       INP XP:WRO(5) , XM:WSET(5);
 61 !       PAR XMIN=1.25 , XMAX=1.75 , YMIN=0. , YMAX=1. ,YS=.3951,PB=.057;
 62 !       ALG Y:Y(5);
 63 !       END
 64 &
 65 !MACRO PICON
 66 !       STV Z:Z(1);
 67 !       INP XM:WLOAD(1),XP:WLOADS(1);
```

```
 68 &                power load range    Pb     reset    steam flow range
 69 !        PAR XMIN= 0.,XMAX=55.,PB= 0.109,TAU= 2.,YMIN=0.0,YMAX=30.;
 70 !        ALG Y:WLOAD(1);
 71 !END
 72 !MACRO PICON
 73 !        STV Z:Z(2);
 74 !        INP XM:HLOAD(2),XP:HLOADS(2);
 75 &                power load range    Pb     reset    steam flow range
 76 !        PAR XMIN= 0.,XMAX=55.,PB=0.108,TAU= 2.,YMIN=0.0,YMAX=30.;
 77 !        ALG Y:WLOAD(2);
 78 !END
 79 !MACRO PICON
 80 !        STV Z:Z(3);
 81 !        INP XM:HLOAD(3),XP:HLOADS(3);
 82 &                power load range    Pb     reset    steam flow range
 83 !        PAR XMIN=0.0,XMAX=15.,PB=0.192,TAU= 2.,YMIN=0.0,YMAX=15.;
 84 !        ALG Y:WLOAD(3);
 85 !END
 86 !MACRO PICON
 87 !        STV Z:Z(4);
 88 !        INP XM:HLOAD(4),XP:HLOADS(4);
 89 &                power load range    Pb     reset    steam flow range
 90 !        PAR XMIN=0.0,XMAX=30.,PB=0.192,TAU= 2.,YMIN=0.0,YMAX=10.;
 91 !        ALG Y:WLOAD(4);
 92 !END
 93 &   Controll inlet flow of sugar
 94        WAC=(HSET(1)-HRO(1))*10.21+
 95       *    (HSET(2)-HRO(2))*10.27+
 96       *    (HSET(3)-HRO(3))*7.526+
 97       *    (HSET(4)-HRO(4))*5.237+
 98       *    (HSET(5)-HRO(5))*2.648
 99 &
100 !MACRO PICON
101 !STV     Z:SZ;
102 !ALG     Y:WFI;
103 !INP     XP:WAC,XM:0.;
104 !PAR     XMIN=  -50.,XMAX=  50.,PB=.03,
105 !        YMIN=    0.,YMAX= 200.,TAU= 1.;
106 !END
107 &
108        RETURN
109        ENTRY REGOUT
110        WRITE(10,*) ' '
111        WRITE(10,*) ' REG'
112        WRITE(10,9300) 'HSET',HSET
113        WRITE(10,9300) 'HRO',HRO
114        WRITE(10,9300) 'Y',Y
115        WRITE(10,9300) 'Z',Z
116        WRITE(10,9300) 'WAC,WFI & SZ',WAC,WFI,SZ
117        WRITE(10,9300) 'PV (bar)',PV
118        WRITE(10,9300) 'PVO (bar)',PVO
119        WRITE(10,9300) 'OP',OP
120        WRITE(10,9300) 'DOWN',DOWN
121        WRITE(10,9300) 'UP',UP
122        WRITE(10,9300) 'TI (*)',TI
123        WRITE(10,9300) 'TT (*)',TT
124        WRITE(10,9300) 'PI (MW)',PI
125        WRITE(10,9300) 'PT (MW)',PT
126        WRITE(10,9300) 'HLOAD (MW)',HLOAD
127        WRITE(10,9300) 'HLOADS (MW)',HLOADS
128        WRITE(10,9300) 'WLOAD (kg/*)',WLOAD
129 &
130        RETURN
131  9300 FORMAT(1X,A12,5G12.5)
132 &
133        END
134 &
```

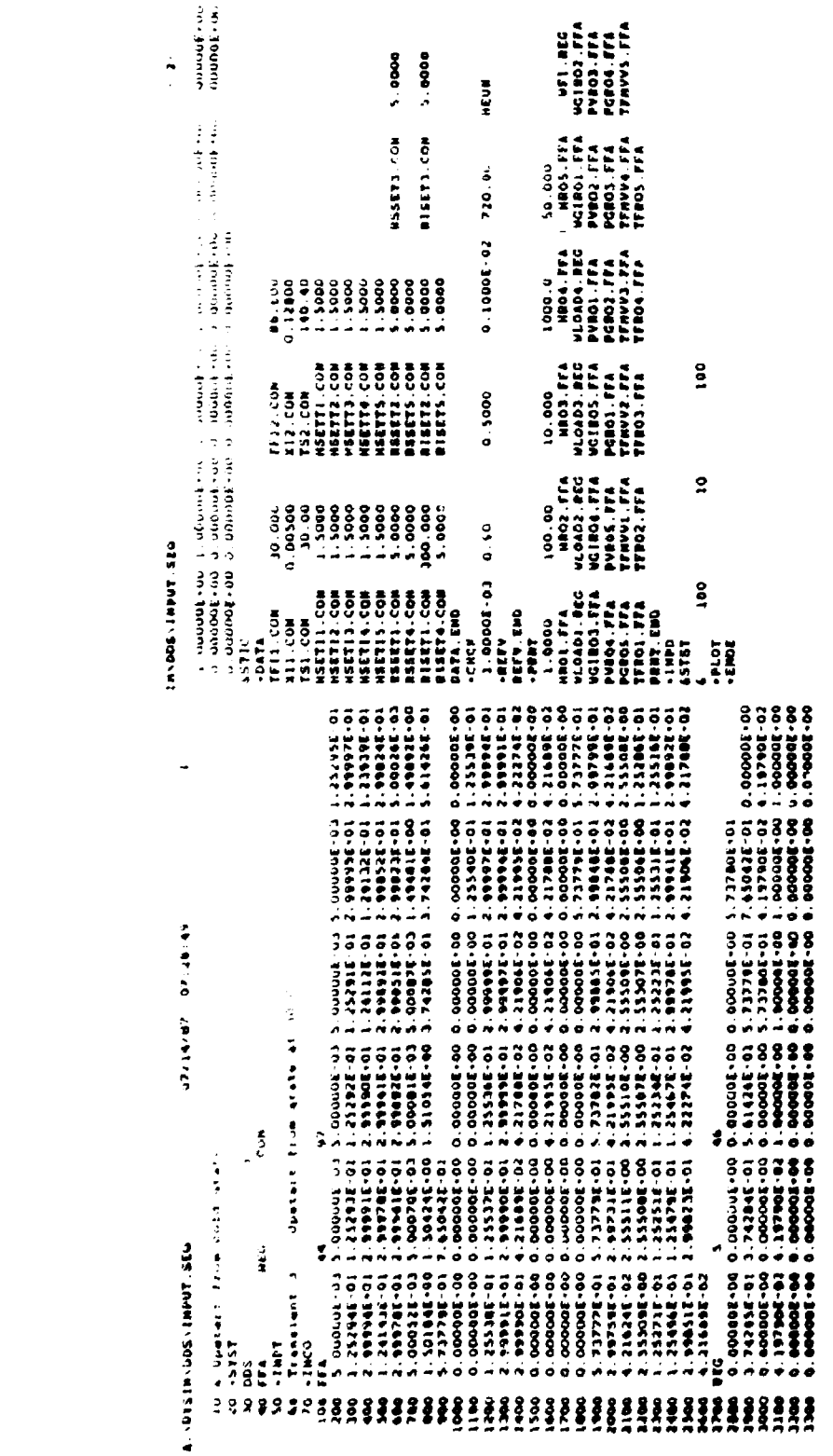
```
A:\DYSIM\DDS\COMSYS.SOU                07/14/87  07:26:34              - 1-
------------------------------------------------------------------------------
  1 ! MNA FFA,REG;
  2 ! PAR   TFI(2),XI(2),TS(2),
  3 !       HSETI(5),HSETT(5),RSSET(5),RISET(5);
  4 &
  5 ! FUN HSAFTT;
  6 &
  7 ! DCL
  8       REAL            TFICON,XICON,HLOADC(4)
  9 &                                   calc. enthalpy load
 10 ! END
 11 &
 12       HLOADC(1)=WLOAD1.REG*HVRO2.FFA
 13       HLOADC(2)=WLOAD2.REG*HVRO3.FFA
 14       HLOADC(3)=WLOAD3.REG*HVRO4.FFA
 15       HLOADC(4)=WLOAD4.REG*HVRO5.FFA
 16 !MOD    REG
 17       HRO1=HRO1.FFA
 18       HRO2=HRO2.FFA
 19       HRO3=HRO3.FFA
 20       HRO4=HRO4.FFA
 21       HRO5=HRO5.FFA
 22       HSET1=HSETI1.CON+AMIN1(1.,T/RSSET1.CON)*(HSETT1.CON-HSETI1.CON)
 23       HSET2=HSETI2.CON+AMIN1(1.,T/RSSET1.CON)*(HSETT2.CON-HSETI2.CON)
 24       HSET3=HSETI3.CON+AMIN1(1.,T/RSSET1.CON)*(HSETT3.CON-HSETI3.CON)
 25       HSET4=HSETI4.CON+AMIN1(1.,T/RSSET1.CON)*(HSETT4.CON-HSETI4.CON)
 26       HSET5=HSETI5.CON+AMIN1(1.,T/RSSET1.CON)*(HSETT5.CON-HSETI5.CON)
 27       HLOAD1=HLOADC(1)
 28       HLOAD2=HLOADC(2)
 29       HLOAD3=HLOADC(3)
 30       HLOAD4=HLOADC(4)
 31       PV1=PVRO2.FFA
 32       PV2=PVRO3.FFA
 33       PV3=PVRO4.FFA
 34       PV4=PVRO5.FFA
 35 !END
 36 &
 37 ! MOD FFA
 38       TFICON=TFI1.CON+AMIN1(1.,T/RISET2.CON)*(TFI2.CON-TFI1.CON)
 39       XICON=XI1.CON+AMIN1(1.,T/RISET2.CON)*(XI2.CON-XI1.CON)
 40       TFI=TFICON
 41       XI =XICON
 42       HFI=HSAFTT(TFICON,XICON)
 43       PSTEAM=PSATF(TS1.CON+AMIN1(1.,T/RISET1.CON)*(TS2.CON-TS1.CON))
 44       WFI=WFI.REG
 45       WLOAD1=0.
 46       WLOAD2=WLOAD1.REG
 47       WLOAD3=WLOAD2.REG
 48       WLOAD4=WLOAD3.REG
 49       WLOAD5=WLOAD4.REG
 50       Y1=Y1.REG
 51       Y2=Y2.REG
 52       Y3=Y3.REG
 53       Y4=Y4.REG
 54       Y5=Y5.REG
 55 ! END
 56 &
 57 ! OUT
 58       WRITE(10,'(A)') '1'//' '
 59       WRITE(10,'(A)') ' Profile'
 60       CALL FFAOUT
 61       CALL REGOUT
 62 &
 63  9300 FORMAT(1X,A12,5G12.5)
```

APPENDIX I

LISTING OF THE LIST FILE THAT WAS USED FOR SIMULATION OF A PART OF A SUGAR FACTORY

APPENDIX J

LISTING OF THE INPUT FILE THAT WAS USED FOR SIMULATION OF A PART OF A SUGAR FACTORY



APPENDIX K

DATA FOR THE SIMULATION OF A PART OF A POWER PLANT

The parameters are the same as those that were used in another project (CHRISTENSEN,1986) except for a few minor changes. All of the parameters, except for the set points (X_S) to the PI-controllers and the inlet temperature of seawater (T_{Ci}), were made part of the modules since they were kept constant during all the simulation experiments.

Parameters for the feedwater preheater chain section

Steam coolers

K_{ST}	MW/K	11.55	15.00	15.45	19.56	18.66
K_{TF}	MW/K	5.775	7.500	7.725	9.780	9.330
C_T	MJ/K	3.02	6.12	7.26	8.47	8.47
C_I,C_O	MJ/K	1	2	2.34	2.14	2.05
C_F	MJ/K	10	20	23.4	21.4	20.5
C_{PF}	kJ/kg·K	4.2	4.2	4.2	4.22	4.28
C_{PD}	kJ/kg·K	4.18	4.19	4.21	4.25	4.34
RHOD	kg/m^3	982	962	949	916	882
AD	m^2	6.28	4.54	9.04	4.02	4.02
τ	s	0.25	0.25	0.25	----	----
RG	kg/m^3·bar	----	----	----	0.47	0.47
V	m^3	----	----	----	80	80

PI-controllers

X_O	m	0.9	0	0	0.394	0.394
X_M	m	2.6	1.4	1.35	1.194	1.194
X_S	m	1.75	0.42	0.405	0.794	0.794
P_B		0.5	0.7	0.5	1.0	1.0
τ_I	s	360	38.4	90	180	19.2
G		1.25	2.5	1.25	2.50	1.25

Heat exchangers

K (1)	MW/K	3.74	4.24	4.14	2.08
C_T	MJ/K	2.8	1.56	3.14	1.29
C_W	MJ/K	84	48	52	28
C (2)	MJ/K	0.922	0.506	0.800	0.312
C_C	MJ/K	9.22	5.06	8.00	3.12
C_{PW}	kJ/kg·K	4.19	4.21	4.25	4.34
C_{PC}	kJ/kg·K	4.2	4.2	4.2	4.22

(1) $K = K_{WT} = K_{TC}$
(2) $C = C_{CI} = C_{CO}$

Delay tubes

L	m^3	26.8	9.84	16.74	3.66	15.6
RHOF	kg/m^3	982.6	965.8	952.2	920.2	888.7

Valve constants

CV	kg/s	630	630	800	400	210

Parameters for the turbine section

HP-turbine and steam line

Flow coefficients

$k_e = 3.31 \cdot 10^{-6}$ bar/(kg^2/s^2)
$K_v = 24.86$ kg/s·bar
$K_h = 13.00$ kg/s·bar

Volume $V_h = 10$ m^3

$\partial\rho/\partial P$ $\rho'_{gv} = 0.54$ kg/m^3·bar
$\rho'_{gh} = 0.51$ kg/m^3·bar

Efficiency $\eta_h = 0.781$

Reheater

Flow coefficients

$K_{hl} = 1630$ kg/s·bar
$K_{vm} = 150$ kg/s·bar

Volumes $V_R = 240$ m^3
$V_T = 284$ m^3
$V_{MO} = 18.6$ m^3

$\partial\rho/\partial P$ $\rho'_{gr} = 0.47$ kg/m^3·bar

Heat transfer

C_{kgr} = 294 (kg/s) / (MW/K)
k_{gt} = 4.54 MW/K
C_{tr} = 5.315 MJ/K

LP-turbine

Flow coefficient

K_l = 121.35 kg/s·bar

Volume V_l = 156 m^3

$\partial\rho/\partial P$ ρ'_{gl} = 0.425 kg/m^3·bar

Efficiency γ_h = 0.87867

Condenser

Heat transfer

k_{cw} = 50 MW/K
c_{pc} = 4.20 kJ/kg·K
C_{cw} = 446 MJ/K
C_{ct} = 73 MJ/K
B_c = 0.455
T_{ci} = 8°C

"Heater model"

Time constant

τ = 10 s

APPENDIX L

LISTING OF SOME FORTRAN77 FILES PRODUCED BY THE PRECOMPILER SYSTEM

Here FORTRAN77 files produced by the precompiler system for the simulation of a part of a sugar factory are listed. These FORTRAN77 files can be compared with the source files listed in Appendix H to see the difference and also what the user has been relieved from.

```
A:\DTSIM\DDS\BEG.FOR                    07/16/87  07:33:36            - 1-
      1       SUBROUTINE BEG(T,NB)
      2 C *
      3       REAL          T
      4       INTEGER       NB
      5 C *
      6       REAL
      7      S Z(4), SZ,
      8      D DBEG( 5),
      9      A Y(5), WLOAD(4), WFI, PVO(4), OP(4), DOWN(4), UP(4), TI(4), TT(4),
     10      A PI(4), PT(4), HLOADS(4),
     11      I NRO(5), NSET(5), HLOAD(4), PV(4)
     12 C *
     13       COMMON /CBEG/
     14      S Z, SZ,
     15      D DBEG,
     16      A Y, WLOAD, WFI, PVO, OP, DOWN, UP, TI, TT, PI, PT, HLOADS,
     17      I NRO, NSET, HLOAD, PV
     18 C *
     19 C *
     20       SAVE
     21 C *
     22       REAL VAC
     23       CALL RAMPSW(NB,
     24      P0.1000E-01, 1.500      , 1.600     ,0.0000     , 41.58
     25      P 1.000     ,
     26      A PVO(1), OP(1), DOWN(1), UP(1), TI(1), TT(1), PI(1), PT(1),
     27      I HLOADS(1), PV(1), T)
     28       CALL RAMPSW(NB,
     29      P0.1000E-01, 1.300      , 1.400     ,0.0000     , 41.21     ,
     30      P 1.000     ,
     31      A PVO(2), OP(2), DOWN(2), UP(2), TI(2), TT(2), PI(2), PT(2),
     32      I HLOADS(2), PV(2), T)
     33       CALL RAMPSW(NB,
     34      P0.1000E-01,0.9000      , 1.000     ,0.0000     , 7.336     ,
     35      P0.5000     ,
     36      A PVO(3), OP(3), DOWN(3), UP(3), TI(3), TT(3), PI(3), PT(3),
     37      I HLOADS(3), PV(3), T)
     38       CALL RAMPSW(NB,
     39      P0.1000E-01,0.2000      ,0.3000     ,0.0000     , 16.54     ,
     40      P0.5000     ,
     41      A PVO(4), OP(4), DOWN(4), UP(4), TI(4), TT(4), PI(4), PT(4),
     42      I HLOADS(4), PV(4), T)
     43       HLOADS(1)=PI(1)+AMIN1(1.,((T-TI(1))/(TT(1)-TI(1))))*(PT(1)-PI(1))
     44       HLOADS(2)=PI(2)+AMIN1(1.,((T-TI(2))/(TT(2)-TI(2))))*(PT(2)-PI(2))
     45       HLOADS(3)=PI(3)+AMIN1(1.,((T-TI(3))/(TT(3)-TI(3))))*(PT(3)-PI(3))
     46       HLOADS(4)=PI(4)+AMIN1(1.,((T-TI(4))/(TT(4)-TI(4))))*(PT(4)-PI(4))
     47       CALL PCON  (NB,
     48      P 1.250     , 1.750      0.0000     ,0.7200     , 1.000     ,
     49      P0.1000E-01,
     50      A Y(1),
     51      I NRO(1), NSET(1))
     52       CALL PCON  (NB,
     53      P 1.250     , 1.750     ,0.0000     ,0.4460     , 1.000
     54      P0.1000E-01,
     55      A Y(2),
     56      I NRO(2), NSET(2))
     57       CALL PCON  (NB,
     58      P 1.250     , 1.750     ,0.0000     ,0.6249     , 1.000
     59      P0.2700E-01,
     60      A Y(3),
     61      I NRO(3), NSET(3))
     62       CALL PCON  (NB,
     63      P 1.250     , 1.750     ,0.0000     ,0.3508     , 1.000     ,
     64      P0.3000E-01,
     65      A Y(4),
     66      I NRO(4), NSET(4))
     67       CALL PCON  (NB,
     68      P 1.250     , 1.750     ,0.0000     ,0.3951     , 1.000
```

```
      P0.5700E-01,
      A Y(5),
      I HRO(5), HSET(5))
       CALL PICON (NR,
      P0.0000     , 55.00     ,0.1090     , 2.000     ,0.0000     ,
      P 30.00     ,
      S Z(1),
      D DREG(1),
      A WLOAD(1),
      I HLOADS(1), HLOAD(1))
       CALL PICON (NR,
      P0.0000     , 55.00     ,0.1080     , 2.000     ,0.0000     ,
      P 30.00     ,
      S Z(2),
      D DREG(2),
      A WLOAD(2),
      I HLOADS(2), HLOAD(2))
       CALL PICON (NR,
      P0.0000     , 15.00     ,0.1920     , 2.000     ,0.0000     ,
      P 15.00     ,
      S Z(3),
      D DREG(3),
      A WLOAD(3),
      I HLOADS(3), HLOAD(3))
       CALL PICON (NR,
      P0.0000     , 30.00     ,0.1920     , 2.000     ,0.0000     ,
      P 10.00     ,
      S Z(4),
      D DREG(4),
      A WLOAD(4),
      I HLOADS(4), HLOAD(4))
       WAC=(HSET(1)-HRO(1))*10.21*
     *     (HSET(2)-HRO(2))*10.27*
     *     (HSET(3)-HRO(3))*7.526*
     *     (HSET(4)-HRO(4))*5.237*
     *     (HSET(5)-HRO(5))*2.648
       CALL PICON (NR,
      P-50.00     , 50.00     ,0.3000E-01, 1.000     ,0.0000     ,
      P 200.0     ,
      S SZ,
      D DREG(5),
      A WFI,
      I WAC, 0.)
       RETURN
       ENTRY REGOUT
       WRITE(10,*) ' '
       WRITE(10,*) ' REG'
       WRITE(10,9300) 'HSET',HSET
       WRITE(10,9300) 'HRO',HRO
       WRITE(10,9300) 'Y',Y
       WRITE(10,9300) 'Z',Z
       WRITE(10,9300) 'WAC,WFI & SZ',WAC,WFI,SZ
       WRITE(10,9300) 'PV (bar)',PV
       WRITE(10,9300) 'PVO (bar)',PVO
       WRITE(10,9300) 'OP',OP
       WRITE(10,9300) 'DOWN',DOWN
       WRITE(10,9300) 'UP',UP
       WRITE(10,9300) 'TI (s)',TI
       WRITE(10,9300) 'TT (s)',TT
       WRITE(10,9300) 'PI (MW)',PI
       WRITE(10,9300) 'PT (MW)',PT
       WRITE(10,9300) 'HLOAD (MW)',HLOAD
       WRITE(10,9300) 'HLOADS (MW)',HLOADS
       WRITE(10,9300) 'WLOAD (kg/s)',WLOAD
       RETURN
 9300  FORMAT(1X,A12,5G12.5)
       END
```

```
      SUBROUTINE CON(NR)
*C*
      INTEGER* NR,I
      REAL     T
*C*
*C*
      INTEGER     NFFAS,NFFAA,NFFAX,NFFAP
      PARAMETER (NFFAS=46,NFFAA=97,NFFAX=15,NFFAP=0)
      REAL SFFA(NFFAS),DFFA(NFFAS),AFFA(NFFAA),XFFA(NFFAX)
      REAL SVFFA(NFFAS),DEFFA(NFFAS),AVFFA(NFFAA)
      COMMON /CFFA/  SFFA,DFFA,AFFA,XFFA
*C*
      INTEGER     NREGS,NREGA,NREGX,NREGP
      PARAMETER (NREGS=5,NREGA=46,NREGX=18,NREGP=0)
      REAL SREG(NREGS),DREG(NREGS),AREG(NREGA),XREG(NREGX)
      REAL SVREG(NREGS),DEREG(NREGS),AVREG(NREGA)
      COMMON /CREG/  SREG,DREG,AREG,XREG
*C*
      INTEGER     NCONS,NCONA,NCONX,NCONP
      PARAMETER (NCONS=0,NCONA=0,NCONX=0,NCONP=26)
      REAL PCON(NCONP)
      REAL PACON(NCONP)
*C*
      COMMON /INTVAR/ T,SVFFA,SVREG
      COMMON /DERIV /   DEFFA,DEREG
      COMMON /ALGVAR/   AVFFA,AVREG
      COMMON /DATA  /   PACON
*C*
      REAL            TFICON,XICON,HLOADC(4)
*C*
      SAVE
*C*
*C*
*     TRANSFER STATE VARIABLES FROM DYSIM TO THE MODULES
      DO 201 I=1,NFFAS
```



```
  201   SFFA(I)=SVFFA(I)
      DO 202 I=1,NREGS
  202   SREG(I)=SVREG(I)
*C*
      HLOADC(1)=AREG(6)*AFFA(74)
      HLOADC(2)=AREG(7)*AFFA(75)
      HLOADC(3)=AREG(8)*AFFA(76)
      HLOADC(4)=AREG(9)*AFFA(77)
*C*
       XREG(1)=SFFA(35)
       XREG(2)=SFFA(36)
       XREG(3)=SFFA(37)
       XREG(4)=SFFA(38)
       XREG(5)=SFFA(39)
       XREG(6)=PCON(7)*AMIN1(1.,T/PCON(17))*(PCON(12)-PCON(7))
       XREG(7)=PCON(8)*AMIN1(1.,T/PCON(17))*(PCON(13)-PCON(8))
       XREG(8)=PCON(9)*AMIN1(1.,T/PCON(17))*(PCON(14)-PCON(9))
       XREG(9)=PCON(10)*AMIN1(1.,T/PCON(17))*(PCON(15)-PCON(10))
       XREG(10)=PCON(11)*AMIN1(1.,T/PCON(17))*(PCON(16)-PCON(11))
       XREG(11)=HLOADC(1)
       XREG(12)=HLOADC(2)
       XREG(13)=HLOADC(3)
       XREG(14)=HLOADC(4)
       XREG(15)=AFFA(64)
       XREG(16)=AFFA(65)
       XREG(17)=AFFA(66)
       XREG(18)=AFFA(67)
      CALL REG(T,NR)
*C*
*C*
      TFICON=PCON(1)*AMIN1(1.,T/PCON(23))*(PCON(2)-PCON(1))
      XICON=PCON(3)*AMIN1(1.,T/PCON(23))*(PCON(4)-PCON(3))
       XFFA(2)=TFICON
       XFFA(4)=XICON
       XFFA(3)=HSAFTT(TFICON,XICON)
```

```
      XFFA(6)=PSAIF(PCON(6)+AMIN1(1.,T/PCON(2))*(PCON(6)-PCON(
      XFFA(7)=AREG(16)
      XFFA(8)=
    *   9
      XFFA(7)=AREG(6)
      XFFA(8)=AREG(7)
      XFFA(9)=AREG(8)
      XFFA(10)=AREG(9)
      XFFA(11)=AREG(1)
      XFFA(12)=AREG(2)
      XFFA(13)=AREG(3)
      XFFA(14)=AREG(4)
      XFFA(15)=AREG(5)
      CALL FFA(T,NR)
*C*
*C*
*     TRANSFER STATE VARIABLES AND DERIVATIVES FROM THE MODULES TO DYSIM
*C*
      DO 501 I=1,NFFAS
        SUFFA(I)=SFFA(I)
  501   DEFFA(I)=DFFA(I)
*C*
      DO 502 I=1,NREGS
        SUREG(I)=SREG(I)
  502   DEREG(I)=DREG(I)
*C*
*C*
      RETURN
*C*
*C*
*C*
      ENTRY YOUT
*C*
      WRITE(10,'(A)') '1'//' '
      WRITE(10,'(A)') ' Profile'
```



```
      CALL FFAOUT
      CALL REGOUT
 9300 FORMAT(1X,A12,5G12.5)
*C*
      RETURN
*C*
*C*
      ENTRY DYNAMS
      RETURN
*C*
*C*
      ENTRY ALVAR(NR)
*C*
      IF(NR.EQ.0) THEN
*     TRANSFER PARAMETERS TO THE MODULES
        DO 703 I=1,NCONP
  703     PCON(I)=PACON(I)
      ELSE IF(NR.EQ.1) THEN
*     TRANSFER ALGEBRAIC VARIABLES FROM DYSIM TO THE MODULES
        DO 801 I=1,NFFAA
  801     AFFA(I)=AVFFA(I)
        DO 802 I=1,NREGA
  802     AREG(I)=AVREG(I)
      ELSE IF(NR.EQ.2) THEN
*     TRANSFER ALGEBRAIC VARIABLES FROM THE MODULES TO DYSIM
        DO 901 I=1,NFFAA
  901     AVFFA(I)=AFFA(I)
        DO 902 I=1,NREGA
  902     AVREG(I)=AREG(I)
      END IF
*C*
      RETURN
*C*
      END
*C*
```

APPENDIX M

A COPY OF THE PAPER PRESENTED AT SCSC'87

A paper describing some of the work with the precompiler system in connection with DYSIM was presented at the 1987 Summer Computer Simulation Conference held July 27-30 in Montreal, Quebec, Canada. It is copied below from the proceedings (KOFOED, 1987).

Description of a two-level modular simulation tool for DYSIM

J.E. Kofoed
Department of Energy Technology
Risø National Laboratory
DK-4000 Roskilde, Denmark

ABSTRACT

A simulation tool that assists the user when constructing models is described.

With this tool the model is compiled into a FORTRAN77 code which must be compiled and linked to the simulation package DYSIM86 which performs the simulation and stores data for prints and plots.

The simulation tool is based on a two-level modular approach. A library is available when working on the first level, namely the construction of a module. A module contains the model of a delimited section of a process plant. The module is made from submodules which are taken from the library. The submodules contain models of components such as heat exchangers. At the second level, the modules are assembled to a total model of the plant. Individual testing of the modules makes it easier to test the total model.

The writing and assembling of modules is accomplished using a mixture of precompiler commands and FORTRAN77 statements, and the two levels are reflected in the two precompilers in the simulation tool. The library is completely open to the user, making it possible to extend the library with new submodules. The precompiler system consists of the two precompilers and the routine for processing the library.

INTRODUCTION

At Risø National Laboratory a simulation system DYSIM for continuous simulation was developed. Initially it was merely a routine for solving differential equations written in FORTRAN77. Soon it was obvious that a modular structure should be implemented, as this would facilitate the development of the programme by dividing the model into smaller pieces and allowing individual testing before assembling the modules (Christensen et al. 1986). On this basis a simulation tool has been developed. The concept of modularity was extended to yet another level where models of components can be used. These are in a library as submodules, so that it is feasible to use well-tested models in new simulations.

The simulation tool to be described here converts source codes into FORTRAN77 codes that are linked to the modular version of DYSIM86. The two-level modular approach is applied to the two precompilers, which together with a routine to process the library constitute the simulation tool that will be referred to as the precompiler system. When using the precompiler system the user is relieved of the tedious and error-prone work of administering the communicating through variables.

It was not intended to create a simulation language, on the contrary it was felt to be important to have full access to the standard FORTRAN77 programming language. In this way it is possible to use the elements making up a flexible language. Today only a few simulation languages have this, e.g. ESL (Crosbie et al. 1985).

The development started on a mainframe computer but the improved computing power, the graphics capacity and the low cost of PCs initiated the work of also modifying the simulation system to a PC-version where interactive control of the simulation is possible.

CONCEPT

DYSIM86 is a routine that administers the simulation and prepares data for making print-outs and plots. A modular approach is applied and this is reflected in an input file and a list file as well as in the model of the process plant, (see Fig. 1). The list file contains a list of the variables used in the model, and the input file contains the information that determines how the simulation is to be performed.

The basic concept of the precompiler system is the use of a two-level modular approach. The two levels are:

1) assembling submodules in a module
2) assembling the modules in the connecting system.

The precompiler system produces the FORTRAN77 codes that appear in Fig. 1 and must be linked to DYSIM86. The list file is also produced.

Figure 2 shows schematically how the precompiler system is used and the files that are produced. It is seen at each stage what files are needed, and the user can operate the precompiler system from every stage as long as all the files needed have been previously made.

EXAMPLE

The use of the precompiler system is illustrated by a very simple example that will be described here. We will model a tank from which an amount of hot water is drawn. The inlet flow and the heat supply are regulated to maintain a given level and temperature. This is shown schematically in Fig. 3.

Stating the mass balance you get

$$\dot{M} = W_i - W_o \qquad (1)$$

We state the energy balance equation:

$$(c_p MT)^{\cdot} = Q + c_p(W_i T_i - W_o T) \qquad (2)$$

where

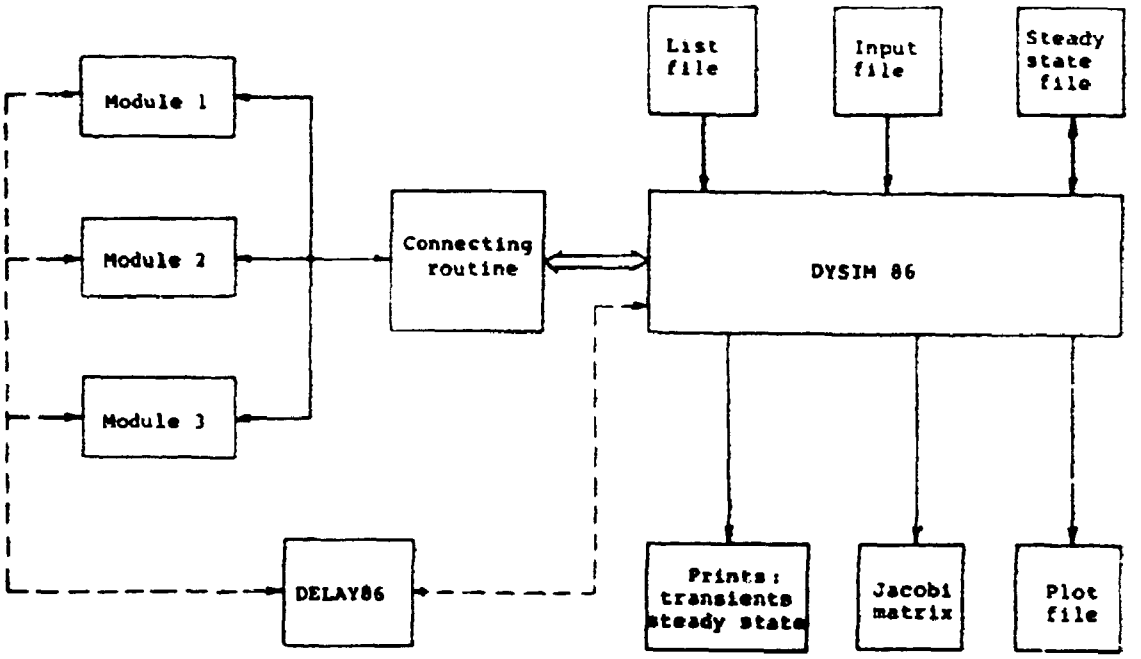


Fig. 1. Schematic description of DYSIM.

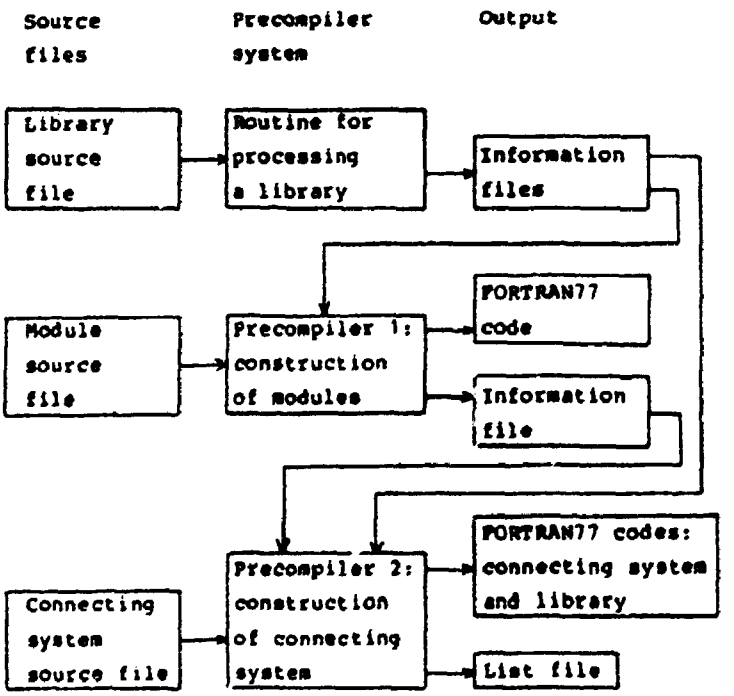


Fig. 2. The precompiler system.

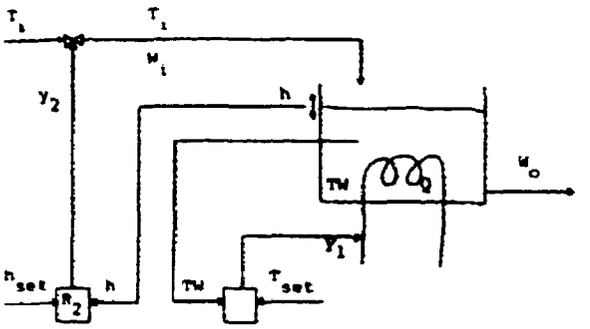


Fig. 3. The controlled tank.

c_p is the specific heat capacity

T is the temperature of water in the tank

T_i is the inlet temperature

Q is the heat flow from the supply

Since

$$(C_p\dot{M}T) = c_p\dot{M}T + c_p\dot{M}\dot{T} = c_p(W_i - W_o)T + c_p\dot{M}\dot{T} \qquad (3)$$

by using Eq. (1), one gets upon insertion into (2)

$$c_p\dot{M}\dot{T} = Q + c_pW_i(T_i - T) \qquad (4)$$

The relation between M and h is

$$h = \frac{M}{A\rho} \qquad (5)$$

where
h is the water level
A is the tank area
ρ is the mass density

Two PI-controllers are used to control heat supply and the inlet water flow. Since a library with a PI-controller is available they are not modelled.

The system will be divided into two modules, one with the tank and one with the controllers.

THE LIBRARY

Functions and submodules that can assist the user are placed in the library. Functions can be used at every level: in submodules, in modules and in the connecting system. The syntax of the functions is that of standard FORTRAN77. The functions are placed before submodules in the library to facilitate the search for functions.

Concerning the syntax the submodules consist of two parts. At the beginning a few commands are used to name the submodule and the variables that are passed to and from the submodule. Commands are lines with an exclamation mark in column 1. The model itself is written using FORTRAN77 except when writing differential equations which will be described shortly. The library that is used in the simple example is shown in Fig. 4.

The SUBMO-command names the submodule and the FUN-command contains those functions that are used in the submodule. This provides the user with a check of the existence of these functions in the library and later they will be picked out if the submodule is used.

```
I:\DYSIM\OPS\SUBMOLIB.SOU                    02/26/87
------------------------------------------------------
      1       FUNCTION ALIM(XMIN, Y, XMAX)
      2       REAL ALIM, X, XMAX, XMIN
      3       ALIM=AMAX1(XMIN,AMIN1(X,XMAX))
      4       RETURN
      5       END
      6 CCCCCC      ======
      7       FUNCTION ALIMD(Y,XMIN,X,XMAX)
      8       IF((X.GT.XMIN) .AND. (X.LT.XMAX))
      9      #  THEN
     10            Z=Y
     11       ELSE IF(
     12      #  ((X.LE.XMIN) .AND. (Y.LT.0.))
     13      #  .OR.
     14      #  ((X.GE.XMAX) .AND. (Y.GT.0.)))
     15      #  THEN
     16            Z=0.
     17       ELSE
     18            Z=Y
     19       END IF
     20       ALIMD=Z
     21       RETURN
     22       END
     23 #
     24 #     ------------------------------
     25 C #
     26 '     SUBMO              PICON
     27 #     =====              =====
     28 'FUN    ALIM,ALIMD;
     29 'STV    Z;
     30 'INP    XP,XM;
     31 'PAR    XMIN,XMAX,PB,TAU,YMIN,YMAX;
     32 'ALG Y;
     33       REAL UP,UM,VP,VM,EPSLON,RANGE
     34 C #
     35 C #    PI-CONTROLLER
     36 C #
     37       RANGE=XMAX-XMIN
     38       UP=(XP-XMIN)/RANGE
     39       UM=(XM-XMIN)/RANGE
     40       VP=ALIM(0.,UP,1.)
     41       VM=ALIM(0.,UM,1.)
     42       EPSLON=(VP-VM)/PB
     43       Z=ALIM(YMIN,Z,YMAX)
     44       Z=ALIMD(EPSLON/TAU,YMIN,Z,YMAX)
     45       Y=ALIM(YMIN,EPSLON+Z,YMAX)
     46       RETURN
     47       END
     48 #
```

Fig. 4. A listing of a simple library.

The four type-giving commands STV, ALG, INP, and PAR give the names of the variables that are passed to and from the submodule as well as the type. This is a great help when looking for submodules that will fit together since it is easy to see whether an output variable in one submodule is connected to an input variable in another. Two more commands are available: TXT makes it possible to transfer a string variable, and UNI makes each implementation of the submodule unique. These two commands are rarely necessary.

The rest of the submodule is written in FORTRAN except for differential equations. In a differential equation one assigns the expression giving the time derivative of a state variable directly to the variable. To distinguish it from an ordinary assignment statement, which is allowed and in some cases necessary, a colon is used instead of an equal sign. The precompiler will construct a variable to hold the value of the derivative and will handle the transfer out of the submodule.

MODULES

The model of a plant can often be broken down into several modules each describing a section, e.g. reactor, turbines, and feedwater preheater chain. By doing this each module can be tested individually prior to making a model of the total plant.

The model of a section can be programmed in a module by using submodules. If a component is not modelled as a submodule it can be programmed directly by ordinary FORTRAN77 statements and differential equations. A submodule is implemented with a MACRO-command which is translated to a CALL-statement, i.e. the statements are not inserted directly in the module.

The head of the module follows almost exactly the syntax applied for a submodule. The only exceptions are the command naming the module and the fact that arrays can be used. The syntax of differential equations is as in submodules.

The two modules in the example of a controlled tank are listed in Fig. 5.

The tank is modelled in TNK and controllers in REG. TNK is an example where the entire model is written using only FORTRAN77 statements and differential equations. Parameters describing the tank with valves and heaters are written in the module since they are not going to be changed. If they were they should be transferred as input variables or input parameters.

In REG the two controllers are implemented with MACRO-commands which start with a subcommand that gives the name of the submodule and is terminated with and END-subcommand. Between these a series of subcommands take care of the transfer of variables between module and submodule. There is one subcommand for each type-giving command in the head of the submodule. The connection between a variable in the submodule and the corresponding variable in the module is established by writing them as a pair separated by a colon with the submodule variable as the first one. The connections are terminated by a comma except for the last connection in a subcommand where a semicolon is used. Parameters are transferred as reals and therefore an equal sign is used.

CONNECTING SYSTEM

DYSIM86 calls the connecting system when the derivatives must be calculated. The connecting system then directs the control to the modules taking care of transport of variables to and from the runtime executive and the modules. All the user has to do is to specify the system under investigation and to write the expressions giving the input variables to the modules. By using a few more commands some additional facilities are available such as using parameters given at runtime in an input file and specifying those functions from the library that are being used in the connecting system.

A:DYSIM\OPG\TNK.SOU 02/26/87

```
1  ' MODUL  TNK
2  ' STV M,TW;
3  ' INP  TI,V1,V2,WD;
4  ' ALG M;
5  &
6         REAL RHO,CP,A,K1,K2
7         DATA RHO/1000./, CP/4.2E3/, A/1./
8         DATA K1/1.E7/, K2/100./
9  &
10        Q=K1*V1
11        WI=K2*V2
12        M : WI-WD
13        TW: (Q*CP*W]*(TI-TW))/(CP*M)
14        H=M/(A*RHO)
15 &
16        RETURN
17        END
```

A:DYSIM\OPG\REG.SOU 02/26/87

```
1  ' MODUL  REG
2  ' STV Z(2);
3  ' ALG Y(2);
4  ' INP M,MSET,TSET,TW;
5  &
6  'MACRO PICON
7  ' STV Z:Z(2);
8  ' ALG Y:Y(2);
9  ' INP XP:MSET, XM:M;
10 ' PAR XMIN=0.8, XMAX=1.2, PB=0.75,
11 '     TAU=1,  YMIN=0,  YMAX=1;
12 'END
13 &
14 'MACRO PICON
15 ' STV Z:Z(1);
16 ' ALG Y:Y(1);
17 ' INP XP:TSET, XM:TW;
18 ' PAR XMIN=40.,  XMAX=60.,  PB=0.50,
19 '     TAU=2,  YMIN=0,  YMAX=1;
20 'END
21 &
22        RETURN
23        END
```

Fig. 5. A listing of two modules.

The connecting system used in our example is listed in Fig. 6. The first command gives the names of the modules in the system. Then one type-giving command makes available some parameters used to introduce perturbations in inlet temperature and load of water as well as the setpoint for the controllers.

The connections to the modules are established in MOD-commands. The first subcommand gives the name of the module to which the connections are given, and the END-subcommand terminates the MOD-command. Between these two subcommands the connections are established with assignment statements but the variables are given in a special form. The names used in the modules or in the head of the connecting system are used with an extension which is either the module name or "CON" for connecting system. The extension is not needed for targets in assignment statements in a MOD-command because the module name is given.

```
A:DYSIM\DPG\CONSYS.SOU                    02/26/87
------------------------------------------------------
  1  ' MAIN REG, TNK;
  2  ' PARM   HO(2), TI(2), RISET(2), MSET, TSET;
  3  #
  4  ' MOD REG
  5        MSET=MSET.CON
  6        TSET=TSET.CON
  7        H=H. TNK
  8        TW=TW. TNK
  9  ' END
 10  #
 11  ' MOD TNK
 12        V1=V1. REG
 13        V2=V2. REG
 14        HO=HO1. CON+AMINI(1., T/RISET1. CON)
 15     #              *(HO2. CON-HO1. CON)
 16        TI=TI1. CON+AMINI(1., T/RISET2. CON)
 17     #              *(TI2. CON-TI1. CON)
 18  ' END
```

Fig. 6. A listing of the connecting system.

In the connecting system it is also possible to write differential equations.

CORRECTIONS

If at any time the precompiler system detects a syntax error or does not recognize a command or variable etc., the user gets the opportunity to change the source file interactively. The erroneous line is printed and the part of the line that the precompiler system was unable to process is indicated. If the user can correct the error by changing the indicated part the precompiler system can be instructed to apply the correction. Sometimes it is impossible to correct an error at the line at which it is detected and then the precompiler must be terminated and the source file corrected. In some cases the user will find himself in the situation of wishing to cancel the correction that have been made, and therefore it is possible to retain the uncorrected version of the file.

DISCUSSION

The precompiler system has been used to construct programmes in several cases from small-to-moderate-size models (below 50 state variables). Examples are a feedwater preheater chain and turbines of a power plant and a multiple effect evaporation plant in a sugar factory. It was found that the automatic transfer of variables and the use of symbolic names in the connecting system saved a lot of time, both in the initial phase of programming and when correcting the models. Errors usually occur when a variable is added to the list of variables that are passed to and from a module. To do this, changes must be made in the module, in the connecting system and in the list file and some neglect of updating was often discovered. Now, by using the precompiler system one only needs to make a correct change in the module and then the precompiler system will automatically implement the necessary corrections in the connecting system and the list file.

Also it proved to be easy to reuse models of components as submodules and to replace one submodule with another.

The modular approach makes it hard to perform the sorting of statements as required in the CSSL67-proposals (Strauss et al. 1968), but it is not an insurmountable problem (Crosbie, 1986). In the simulation language built on the CSSL67-proposals this sorting is possible, e.g. (ACSL 1986). The benefits of applying a modular approach are felt to be so many that the automatic sorting can be dispensed with.

The modular approach makes it easy to test each module individually and this facilitates the construction of the total model of the plant. It is felt that the presentation of results and the clarity in general are improved compared to that of e.g. ACSL. Although for small models this is not a serious problem.

If discrete events requires an adjustment of the integration step length, the user is responsible for that, but this can be programmed in a submodule using a facility in DYSIM86.

As noted earlier, the library is open to the user making it possible to extend the library. This is done when a model has been tested as a module and has been validated. All the user has to do to add it to the library is to change the naming command from MODUL to SUBMO and move the module to the library and process it with the precompiler system.

The precompiler system thus appears to have made simulation faster and more reliable and source codes more readable.

REFERENCES

"Advanced Continuous Simulation Language (ACSL) - Reference Manual", 1986. Mitchell Gauthier Associates Inc., Concord, Mass.

Christensen, P. la Cour, Koford, J.E., Larsen, N., 1976. "A Modular Simulation System for Continuous Dynamic Processes". Risø-M-2607. Risø National Laboratory, DK-4000 Roskilde, Denmark.

Crosbie, R.E., Javey, S., Hay, J.L., Pearce, J.G., 1985. "ESL - A new Continuous System Simulation Language". Simulation, May: 242-246.

Crosbie, R., 1986. "Developments in ESL and other CSSL's for the 1990's". In proceedings of the 1986 Summer Computer Simulation Conference (Reno, Nevada, July 28-30), SCS, San Diego, Ca., 313-314.

DANSK RESUMÉ

Dynamisk simulering af procesanlæg ved anvendelse af (analoge eller) digitale computere kan anvendes til for eksempel design af reguleringssystemer og undersøgelse af unormale hændelsesforløb. Dynamisk simulering indebærer

1. opbygning af en matematisk model
2. dataindsamling
3. kodning af model
4. udførelse af den egentlige simulering
5. studium af simuleringsresultater

Punkterne 3-5 indebærer anvendelse af og derfor kendskab til den computer, der anvendes. Det har vist sig, at de, der har brug for simulering, ofte finder tilegnelsen af dette kendskab som en så stor barriere, at man har veget tilbage fra at foretage en simulering og i stedet har søgt at klare sig uden eller har rekvireret de nødvendige simuleringer "uden for huset". Det er klart hensigtsmæssigt, at den, der foretager simuleringen af anlægget, også er den, der kender anlægget: domæneeksperten. Domæneeksperten har derfor brug for et simuleringssystem, som tillader ham at foretage sine simuleringseksperimenter og modelopbygning på en måde, som ikke kræver detaljeret kendskab til den anvendte computer.

Tanken med dette projekt er udviklingen af en grænseflade til et program, der kan benyttes til afvikling af en simuleringsmodel: DYSIM86. Resultatet er udviklingen af et prækompilersystem, der tillader opbygning af en model ved en blanding af standard FORTRAN programafsnit og enkle kommandoer. Der anvendes et modulær koncept i to niveauer.

Det ene niveau er arvet fra DYSIM86, som anvender en opdeling af modellen i større dele. På det laveste niveau anvendes delmodeller for komponenter. Disse delmodeller er placeret i et bibliotek. På denne måde får brugeren adgang til en serie gennemtestede modeller.

Baggrunden for projektet er altså DYSIM86: den rutine, der udfører selve simuleringen. Det udviklede sæt af prækompilere i samspil med DYSIM86 sammenlignes med andre simuleringssystemer og -sprog, ligesom der opridses et par standardiseringsforslag fra litteraturen.

Prækompilersystemet beskrives kort ved at angive den valgte syntaks med en kort begrundelse, men der findes ikke en fuldstændig liste med muligheder og begrænsninger. Disse vil være at finde i en manual.

Når der skal ske en sammenknytning på de to niveauer, skal det gøres på en meningsfuld måde, og dertil er der udviklet en grafisk notation, som beskrives kort.

Anvendelsen af denne notation og af prækompilersystemet illustreres ved to forskellige simuleringer. Den ene simulering er af en del af et kraftværk, hvor turbiner og fødevandsforvarmerkæde modelleres, programmeres og simuleres.

Den anden simulering er af en del af en sukkerfabrik, hvor et flertrinsfordamperanlæg med tilhørende sæt af afspændere og sukkerforvarmere modelleres, programmeres og simuleres. Den modulære koncept anvendes på to forskellige måder i de to eksempler.

Ved sammenligningen mellem vores og andres simuleringssystemer og med standardiseringsforslag viser det sig, at det modulære koncept, og ikke mindst genanvendelsen af delmodeller fra et bibliotek, er ved at vinde en større udbredelse. Med dette arbejde har man altså fået opbygget et moderne simuleringssystem. Styrken ved det modulære koncept illustreres bl.a. ved at påpege, hvorledes en kompletering af de to modeller let kan ske ved at bygge et nyt modul til de eksisterende modeller.

Til slut i rapporten evalueres resultatet af arbejdet i lyset af simuleringssystemet, som det var før prækompilersystemet var udviklet, og i lyset af kommercielt tilgængelige simuleringssystemer. I denne evaluering indgår hensynet til det faktum, at da vi selv har udviklet systemet, er vi bedre i stand til at få opfyldt vore behov nu og ikke mindst i fremtiden ved blot en lille indsats. Endvidere foreslås en del forbedringer til det eksisterende simuleringssystem, og der ridses kort nogle tanker op til et nyt, grafisk baseret system.

Riso National Laboratory

Riso – M – 2652

Title and author(s)

Development of a Two-level Modular Simulation Tool for Dysim

Jan Eggert Kofoed

Date

Department or group

Energy Technology

Groups own registration number(s)

Project/contract no.

Pages 266 Tables 13 Illustrations 69 References 41

ISBN87-550-1330-9

Abstract (Max. 2000 char.)

A simulation tool to assist the user when constructing continuous simulation models is described.

The simulation tool can be used for constructing simulation programmes that are executed with the runtime executive DYSIM86 which applies a modular approach. This approach makes it possible to split a model into several modules. The simulation tool introduces one more level of modularity. In this level a module is constructed from submodules taken from a library. A submodule consists of a submodel for a component in the complete model.

The simulation tool consists of two precompilers working on the two different levels of modularity. The library is completely open to the user so that it is possible to extend it. This is done by a routine which is also part of the simulation tool.

The simulation tool is demonstrated by simulating a part of a power plant and a part of a sugar factory. This illustrates that the precompilers can be used for simulating different types of process plants.

Descriptors INIS

COMPUTERIZED SIMULATION; D CODES; FEEDWATER HEATERS; MATHEMATICAL MODELS; MODULAR STRUCTURES; NUCLEAR POWER PLANTS; STEAM TURBINES; TRANSLATORS.

Available on request from Riso Library, Riso National Laboratory, (Riso Bibliotek, Forskningscenter Riso), P.O. Box 49, DK-4000 Roskilde, Denmark.
Telephone 02 37 12 12, ext. 2262. Telex: 43116, Telefax: 02 36 06 09

Available on request from.
Risø Library,
Risø National Laboratory, P.O. Box 49,
DK-4000 Roskilde, Denmark
Phone (02) 37 12 12 ext.2262

ISBN 87-550-1330-9
ISSN 0418-6435